IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

RIVERBED TECHNOLOGY, INC., Plaintiff, v. QUANTUM CORPORATION, A.C.N. 120 786 012 PTY, LTD, and ROCKSOFT LTD., Defendants.	)	**REDACTED PUBLIC VERSION** C.A. No. 08-016-SLR

DECLARATION OF MAURICIO A. FLORES
IN SUPPORT OF DEFENDANTS' MOTION TO DISMISS

ASHBY & GEDDES
Steven J. Balick (I.D. # 2114)
John G. Day (I.D. #2403)
Tiffany Geyer Lydon (I.D. #3950)
500 Delaware Avenue, 8th Floor
P.O. Box 1150
Wilmington, DE 19899
(302) 654-1888
sbalick@ashby-geddes.com
jday@ashby-geddes.com
tlydon@ashby-geddes.com

Attorneys for Defendants

Of Counsel:

Amar L. Thakur
Mauricio A. Flores
Sheppard, Mullin, Richter & Hampton LLP
12275 El Camino Real, Suite 200
San Diego, CA 92130-2006
(858) 720-8900

Dated: February 14, 2008
188210.1

I, Mauricio A. Flores, state and declare as follows:

1. I am an attorney in the law firm of Sheppard Mullin Richter & Hampton LLP ("SMRH"), counsel of record for Quantum Corporation ("Quantum"). I am over twenty-one years of age and not under any legal disability. I have personal knowledge of the following facts and, if called as a witness, could and would testify competently thereto.

Progress of the '810 and '249 Patent Litigation in the Northern District of California

2. On August 14, 2007, Quantum filed a patent infringement action against Riverbed Technology, Inc. ("Riverbed") in the Northern District of California, entitled *Quantum Corporation v. Riverbed Technology, Inc.*, Case No. C-07-04161-WHA (the "California Action"). The California Action contained one claim for infringement of U.S. Patent No. 5,990,810 (the "'810 Patent"). Quantum served the Summons and Complaint on October 9, 2007. A true and correct copy of this Complaint is attached as Exhibit A.

3. Riverbed answered on October 29, 2007, asserting a counterclaim for a declaratory judgment that the '810 Patent is not infringed and is invalid. The case was reassigned from Magistrate Judge Patricia V. Trumbull to the Honorable William H. Alsup ("Judge Alsup") on October 30, 2007. The next day, the Initial Case Management Conference was set for November 15.

4. Over the next week, the parties completed their ADR disclosures required by the local rules and successfully requested that the Initial Case Management Conference be moved to November 29, 2007. Specifically, on November 2, 2007, the parties engaged in their required Fed. R. Civ. P. 26(f) conference, discussing discovery and case management issues. That same day, Riverbed hand-served Quantum's counsel with Riverbed's First Set of Requests for

Production of Documents and Things (Nos. 1- 76) and First Set of Interrogatories (Nos. 1- 8), directed to discovery relevant to Quantum's claim for infringement of the '810 Patent.

5. On November 13, 2007, Riverbed filed its First Amended Answer and Counterclaims. In addition to its declaratory judgment counterclaims regarding the '810 Patent, Riverbed added an affirmative counterclaim against Quantum based on alleged infringement of U.S. Patent No. 7,116,249 (the "'249 Patent"). A true and correct copy of Riverbed's First Amended Answer and Counterclaims is attached as Exhibit B.

6. On November 14, 2007, Quantum served its First Set of Requests for Production of Documents and Things on Riverbed via hand delivery (Nos. 1-76). These requests were directed to discovery relevant to Quantum's claim for infringement of the '810 Patent.

7. On November 16, 2007, Riverbed served its Fed. R. Civ. P. 26 Initial Disclosures on Quantum. On November 20, Quantum served its Fed. R. Civ. P. 26 Initial Disclosures on Riverbed. The parties filed their Joint Case Management Statement on November 21, 2007.

8. On November 23, 2007, the parties served each other with second rounds of discovery. Riverbed served Quantum by hand delivery with its Second Set of Requests for Production of Documents and Things (Request Nos. 77-179) and Second Set of Interrogatories (Interrogatory Nos. 9-14). These requests were directed to discovery relevant to Riverbed's claim for infringement of the '249 Patent. That same day, Quantum served Riverbed via U.S. Mail with its Second Set of Requests for Production of Documents and Things (Request Nos. 77-146) directed to Riverbed's claim for infringement of the '249 Patent. Quantum also served its First Set of Interrogatories on Riverbed (Nos. 1 -17) directed to Quantum's claim for infringement of the '810 Patent.

9. On November 29, the Court held the Initial Case Management Conference. That same day, the Court entered a scheduling order setting the claim construction hearing on May 7, 2008 and trial for February 2, 2009. A true and correct copy of this scheduling order is attached as Exhibit C.

10. On December 3, 2007, Quantum responded in writing to Riverbed's First Set of Requests for Production of Documents and Things and First Set of Interrogatories and produced documents relating to issues regarding Quantum's claim for infringement of the '810 Patent.

11. On December 7, 2007, pursuant to a deadline set by the Court, the parties exchanged supplemental Fed. R. Civ. P. 26 Initial Disclosures, which the Court had indicated at the Initial Case Management Conference on November 29, 2007 would likely be the parties' final chance to supplement their initial disclosures and that they would likely be held to the witnesses and documents disclosed therein. These disclosures included information the parties gathered with respect to both the '810 Patent and the '249 Patent. That same day, the Court entered an order that had been proposed by the parties after meeting and conferring regarding several discovery-related issues, including privilege logs, expert reports, and depositions. True and correct copies of the Case Management Order and Order re Schedule for Claim Construction are attached as Exhibit D.

12. On December 13, 2007 Quantum served its Preliminary Infringement Contentions on Riverbed pursuant to Northern District of California Patent Local Rule 3-1, along with documents pursuant to Patent Local Rule 3-2(a). These materials all related to Quantum's contentions regarding its affirmative claim for infringement of the '810 Patent. Likewise, Riverbed served its Preliminary Infringement Contentions regarding the '249 Patent along with 1,021 pages of documents.

13. On December 14, 2007, Riverbed served its written responses to Quantum's First Set of Requests for Production of Documents and Things. That same day, the Court entered a Stipulated Protective Order for the handling of discovery materials.

14. On December 17, 2007, Quantum served its Initial Privilege Log. That same day, Riverbed served Quantum with its Third Set of Interrogatories (Interrogatory No. 15) along with a production of documents. This discovery was aimed at obtaining information regarding Quantum's acquisition and analysis of Riverbed products.

15. On December 21, 2007, Quantum served written responses to Riverbed's Second Set of Requests for Production of Documents and Things and Second Set of Interrogatories along with a production of documents.

16. On December 24, 2007, after exchanging meet and confer correspondence and speaking telephonically with opposing counsel, Quantum served Supplemental Preliminary Infringement Contentions regarding infringement of the '810 Patent, as well as its First Set of Supplemental Responses to Riverbed's First Set of Interrogatories (directed to the '810 Patent), along with a production of documents. That same day, Riverbed served its written responses to Quantum's Second Set of Requests for Production of Documents and Things and First Set of Interrogatories.

17. On December 28, 2007, following an order of the Court on a proposal by the parties, Quantum served documents requested to be produced pursuant to Patent Local Rule 3-2(b), which documents related to the conception and reduction to practice of the inventions in the '810 Patent. That same day, Riverbed served its Initial Privilege Log.

18. On January 9, 2008, following meet and confer efforts by the parties, Quantum served Riverbed with its Second Set of Supplemental Responses to Riverbed's First Set of

Interrogatories (directed to the '810 Patent) and its First Set of Supplemental Responses to Riverbed's Second Set of Interrogatories, along with documents. That same day, Quantum sent a letter to Riverbed detailing the deficiencies in Riverbed's responses to Quantum's First Set of Interrogatories and requesting supplemental responses to many individual interrogatories.

19. On January 14, 2008, Quantum filed a motion to compel responses to its First Set of Interrogatories.

20. On January 17, 2008, Quantum produced documents relating to Australian provisional patent applications from which the '810 Patent claims priority, along with other documents relating to Quantum's standing to sue for infringement of the '810 Patent.

21. On January 22, 2008, Quantum served its written responses to Riverbed's Third Set of Interrogatories, along with documents.

22. On January 31, 2008, Quantum filed its Preliminary Invalidity Contentions pertaining to the '249 Patent.

Riverbed's Motion to Dismiss the '810 Patent Claim

23. On January 9, 2008, Riverbed filed a motion to dismiss based on Quantum's purported lack of standing to sue for infringement of the '810 Patent. That same day, Riverbed filed a Declaratory Judgment Action for non-infringement and invalidity of Quantum's '810 Patent with the United States District Court for the District of Delaware.

24. On January 31, 2007, Judge Alsup heard oral argument on the motion to dismiss. I attended this hearing. A true and correct copy of the transcript of this hearing before Judge Alsup is attached as Exhibit E.

25. On February 1, 2008, Riverbed's counsel submitted a letter to Judge Alsup affirming that should the Court grant the motion to dismiss, Riverbed was inclined to maintain its infringement counterclaim for the '249 Patent in the Northern District of California. A true and correct copy of this letter is attached as Exhibit F.

26. On February 4, 2008, Judge Alsup granted Riverbed's motion and dismissed Quantum's '810 infringement action without prejudice. A true and correct copy of Judge Alsup's order is attached as Exhibit G.

27. On February 8, 2008, Quantum, A.C.N. and Rocksoft executed a new licensing agreement which grants Quantum an exclusive license to '810 Patent. A true and correct copy of this agreement is attached as Exhibit H.

28. Quantum has or will be filing a Motion for Leave to File a First Amended Reply to Riverbed's Counterclaim, and Counterclaims with the California court. Pursuant to this motion, Quantum seeks to file a counterclaim for infringement of the '810 Patent.

Contacts with the California

29. On February 11, 2008, I reviewed Riverbed's website. Riverbed's "world headquarters" is listed as San Francisco, California. According to Riverbed's website, it also has offices in New York, NY, Sunnyvale, California, Urbana, Illinois, and in numerous locations in Europe and Asia Pacific. There are no listed office locations in Delaware.

30. Riverbed's Initial Disclosure, entry 1.2, lists five locations which contain documents and other electronically stored information relating to the infringement actions. Three of the locations are in California (San Francisco and Mountain View), the other two are in Illinois and Georgia. None of the locations listed are in Delaware. In its First Supplemental Disclosure,

Riverbed listed an additional location in Redwood Shores, California. Again, no Delaware locations are listed.

31. In its initial disclosures, Riverbed identifies seven Riverbed employees who have information regarding the '249 Patent and the "structure and operation of Riverbed's Steelhead appliances and the Riverbed Optimization Sytem, (RiOS) software platform," Riverbed's accused product under the '810 patent. Since none of Riverbed's offices are in Delaware and none of the sites identified by Riverbed as containing relevant information in its initial disclosures are in Delaware, I believe that none of these witnesses reside in Delaware.

32. Quantum has documents and other electronically stored information relating to the infringement actions in a number of California locations. To the extent of my knowledge, Quantum does not have any documents relating to the '810 claim in Delaware.

33. A number of Quantum employees, identified as witnesses by both Quantum and Riverbed, reside in California. They include: James Hall, Brad Cohen, Anup Pal, Alan Olson, Steve Dalton, Jeffery Tofano. A number of other witnesses reside in Washington state including: Jonathan Otis, Jon Gacek, Steve Whitner, Gabriel Chaher, Bill Brits, and Rob Clark. None of the witnesses identified by Quantum or Riverbed reside in Delaware.

34. This litigation involves highly sensitive source code data. Quantum has agreed that Riverbed may review its source code data in the offices of its counsel, SMRH, located in San Francisco, California. Likewise, Riverbed has agreed that Quantum may review its source code data in the office of its counsel, Quinn Emmanuel, also located in San Francisco, California.

35. I work out of SMRH's Del Mar office, which is located in California. Other SMRH attorneys who represent Quantum in this litigation reside in San Francisco, California and work out of SMRH's San Francisco office. SMRH does not have an office in Delaware.

36. On February 11, 2008, I reviewed Quinn Emmanuel's website, counsel for Riverbed, which lists five different office locations, none of which are Delaware.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct and that this Declaration was executed in San Diego, California on February 13, 2008.

By _____
MAURICIO A. FLORES

EXHIBIT A

Steven E. Bledsoe (CA Bar No. 157811)
ARENT FOX LLP
445 S. Figueroa Street, Suite 3750
Los Angeles, CA 90071-1601
Telephone: 213.629.7400
Facsimile: 213.629.7401
bledsoe.steven@arentfox.com

Attorneys for Plaintiff Quantum Corporation

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

C 07 4161

QUANTUM CORPORATION,

Plaintiff,

v.

RIVERBED TECHNOLOGY, INC.,

Defendant.

Case No.

COMPLAINT

DEMAND FOR JURY TRIAL

Plaintiff Quantum Corporation ("Quantum"), by its undersigned counsel, brings this action against Defendant Riverbed Technology, Inc. ("Riverbed") and alleges as follows:

THE PARTIES

1. Plaintiff Quantum is a Delaware corporation having a principal place of business at 1650 Technology Drive, Suite 700, San Jose, CA 95110.

2. On information and belief, Riverbed is a corporation organized and existing under the laws of the State of Delaware, having a place of business at 199 Fremont Street, San Francisco, CA 94105.

JURISDICTION AND VENUE

3. This action is for patent infringement arising under the Patent Laws of the United States, 35 U.S.C. § 1 *et seq.*, and seeks damages and injunctive relief. This Court has jurisdiction over this action pursuant to 28 U.S.C. §§ 1331 and 1338(a).

4. On information and belief, this Court has personal jurisdiction over Riverbed because Riverbed currently sells products and transacts business within this judicial district, and otherwise transacts business throughout this state.

5. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391 and 1400(b).

BACKGROUND

6. Ross Neil Williams conceived and reduced to practice the invention disclosed and claimed in United States Patent No. 5,990,810 (hereinafter "the '810 patent") directed to a new, useful and non-obvious method and apparatus for partitioning a block of data into subblocks and for storing and communicating such subblocks. The '810 patent complies with all of the requirements of 35 U.S.C. § 1 *et seq.*

7. On November 23, 1999, the United States Patent and Trademark Office (the "PTO") duly and lawfully issued the '810 patent entitled "Method for Partitioning a Block of Data into Subblocks and for Storing and Communicating such Subblocks." A copy of the '810 patent is attached as Exhibit A.

8. Rocksoft Limited, a subsidiary of Quantum, owns all right, title, and interest in the '810 patent. Quantum is an exclusive licensee of the '810 patent.

9. On information and belief, Riverbed has manufactured, sold, offered to sell and/or imported Riverbed Steelhead products that infringe the '810 patent.

COUNT FOR INFRINGEMENT OF U.S. PATENT NO. 5,990,810

10. Quantum hereby realleges and incorporates by reference the allegations of paragraphs 1-9 of this complaint.

11. On information and belief, Riverbed, without authority or license from Plaintiff, has infringed and continues to infringe by direct infringement, contributory infringement, and/or

inducement to infringe, the '810 patent either literally or under the doctrine of equivalents, in violation of 35 U.S.C. § 271, by practicing the method claimed by the '810 patent and by selling, offering to sell or importing products that employ the claimed method in this judicial district and elsewhere throughout the United States.

12. On information and belief, Riverbed, without authority or license from Plaintiff, actively induces others to directly infringe the '810 patent and/or contributorily infringe the '810 patent.

13. On information and belief, Riverbed has committed such acts of infringement with knowledge of the '810 patent.

14. Such acts of infringement by Riverbed have been, and continue to be, willful and deliberate, and Plaintiff believes such acts will continue in the future unless enjoined by this Court.

15. By reason of Riverbed's acts of infringement, Plaintiff has suffered and continues to suffer irreparable harm and damages, including, but not limited to, diminution in the value of the rights granted under the '810 patent.

16. Plaintiff has no adequate remedy at law.

PRAYER FOR RELIEF

WHEREFORE, Quantum prays for judgment in its favor and against Defendant as follows:

A) Entering judgment for Plaintiff that Riverbed directly and/or indirectly infringes the '810 patent;

B) Entering judgment enjoining Riverbed from further infringement of the '810 patent pursuant to 35 U.S.C. § 283;

C) Awarding damages under 35 U.S.C. § 284 including costs and prejudgment interest;

D) Determining that this is an exceptional case under 35 U.S.C. § 285 and awarding Plaintiff its reasonable attorneys' fees, costs and expenses;

E) Awarding Plaintiff treble damages by reason of Riverbed's willful infringement; and

F) Awarding Plaintiff such other relief as the Court deems just and proper.

Dated: August 14, 2007

Respectfully submitted,

By: _____

Steven E. Bledsoe
Arent Fox LLP
445 South Figueroa Street, Suite 3750
Los Angeles, CA 90071-1601
Telephone: 213.629.7400
Facsimile: 213.629.7401

Attorneys for Plaintiff
Quantum Corporation

JURY TRIAL DEMANDED

Plaintiff hereby demands trial by jury.

Dated: August 14, 2007

By: _____

Steven E. Bledsoe
Arent Fox LLP
445 South Figueroa Street, Suite 3750
Los Angeles, CA 90071-1601
Telephone: 213.629.7400
Facsimile: 213.629.7401

Attorneys for Plaintiff
Quantum Corporation

EXHIBIT A

US005990810A

United States Patent [19]

Williams

[11] Patent Number: 5,990,810

[45] Date of Patent: Nov. 23, 1999

[54] METHOD FOR PARTITIONING A BLOCK OF DATA INTO SUBBLOCKS AND FOR STORING AND COMMUNCATING SUCH SUBBLOCKS

[76] Inventor: Ross Neil Williams, 3/305 N. Terrace, Adelaide SA5000, Australia

[21] Appl. No.: 08/894,091

[22] PCT Filed: Feb. 15, 1996

[86] PCT No.: PCT/AU96/00081

§ 371 Date: Aug. 15, 1997

§ 102(e) Date: Aug. 15, 1997

[87] PCT Pub. No.: WO96/25801

PCT Pub. Date: Aug. 22, 1996

[30] Foreign Application Priority Data

Feb. 17, 1995 [AU] Australia PN1232
Apr. 12, 1995 [AU] Australia PN2392

[51] Int. Cl.[6] H03M 7/00

[52] U.S. Cl. 341/51; 341/67

[58] Field of Search 341/51, 50, 67; 375/241; 704/203

[56] References Cited

U.S. PATENT DOCUMENTS

4,698,628	10/1987	Herkert et al.	340/825.02
5,235,623	8/1993	Sugiyama et al.	341/67
5,479,654	12/1995	Squibb	395/600

OTHER PUBLICATIONS

Williams, Ross, "An algorithm for matching text (possibly original)", Newsgroup posting, comp.compression, Jan. 27, 1992.

Williams, Ross, "Parallel data compression", Newsgroup posting, comp.conmpression.research, Jun. 30, 1992.

Knuth, Donald E., "The Art of Computer Programming, vol. 3: Sorting and Searching", pp. 508–513, Addison–Wesley Publishing Company, 1973.

Williams, Ross N., "An Introduction to Digest Algorithms" Proceedings of the Digital Equipment Computer Users Society, pp. 9–18, Aug. 1994.

Williams, Ross N., "An Extremely Fast ZIV–Lempel Data Compression Algorithm", Proceedings of Data Compression Conference, pp. 362–371, Apr. 1991.

Knuth, Donald E., The Art of Computer Programming, vol. 1: Fundamental Algorithms, pp. 435–451, Addison Wesley Publishing Company, 1973.

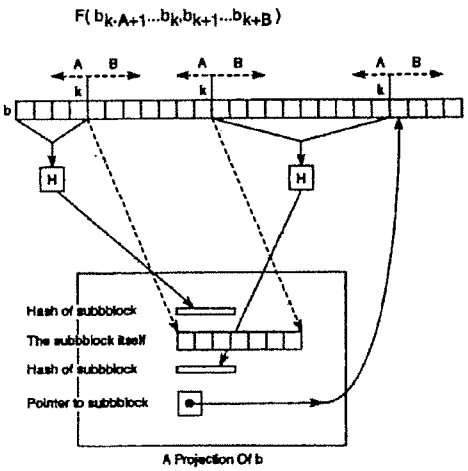
Primary Examiner—Brian Young

Attorney, Agent, or Firm—Greenberg Traurig; Robert P. Bell

[57] ABSTRACT

This invention provides a method and apparatus for detecting common spans within one or more data blocks by partitioning the blocks (FIG. 4) into subblocks and searching the group of subblocks (FIG. 12) (or their corresponding hashes (FIG. 13)) for duplicates. Blocks can be partitioned into subblocks using a variety of methods, including methods that place subblock boundaries at fixed positions (FIG. 3), methods that place subblock boundaries at data-dependent positions (FIG. 3), and methods that yield multiple overlapping subblocks (FIG. 6). By comparing the hashes of subblocks, common spans of one or more blocks can be identified without ever having to compare the blocks or subblocks themselves (FIG. 13). This leads to several applications including an incremental backup system that backs up changes rather than changed files (FIG. 25), a utility that determines the similarities and differences between two files (FIG. 13), a file system that stores each unique subblock at most once (FIG. 26), and a communications system that eliminates the need to transmit subblocks already possessed by the receiver (FIG. 19).

30 Claims, 26 Drawing Sheets



MADD 0003
Sheet 1 of 26

Demonstr	ates con	tent mis	alignmen	t.
XDemonst	rates co	ntent mi	salignme	nt.

Figure 1

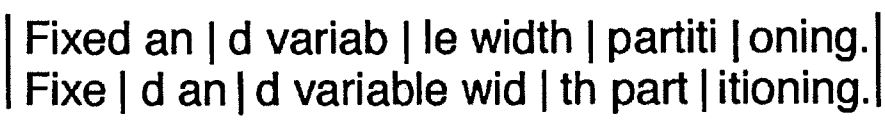

Fixed an | d variab | le width | partiti | oning.
Fixe | d an | d variable wid | th part | itioning.

Figure 2

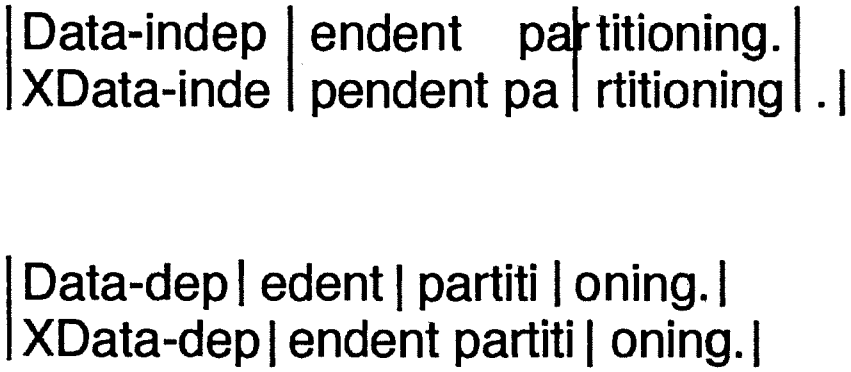

Data-indep | endent pa | rtitioning. |
XData-inde | pendent pa | rtitioning | . |
Data-dep | edent | partiti | oning. |
XData-dep | endent partiti | oning. |

Figure 3

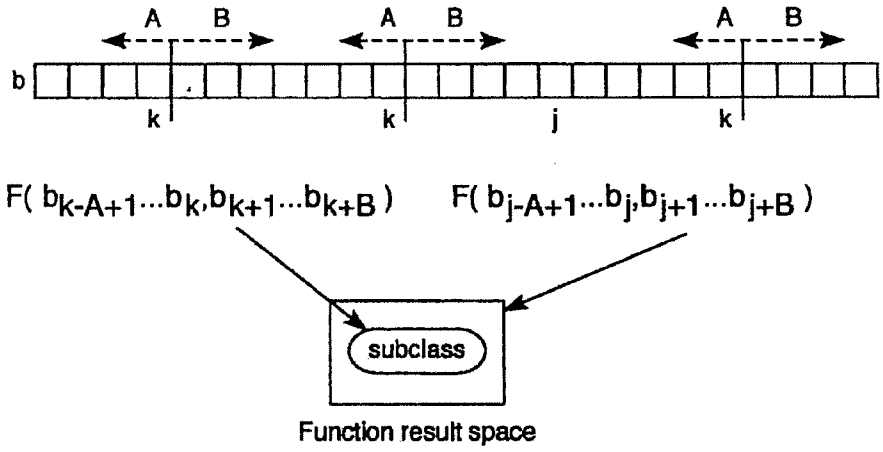

A
B
A
B
A
B
b
k
k
j
k
F($b_{k-A+1}...b_k,b_{k+1}...b_{k+B}$)
F($b_{j-A+1}...b_j,b_{j+1}...b_{j+B}$)
subclass
Function result space

Figure 4

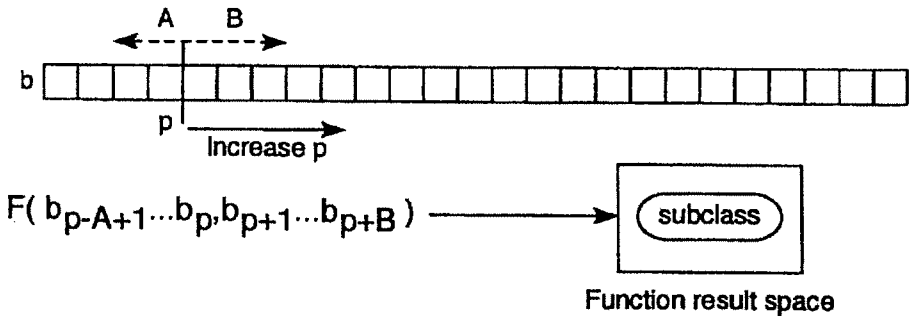

A
B
b
p
Increase p
F(b_{p-A+1}...b_p,b_{p+1}...b_{p+B})
subclass
Function result space

Figure 5

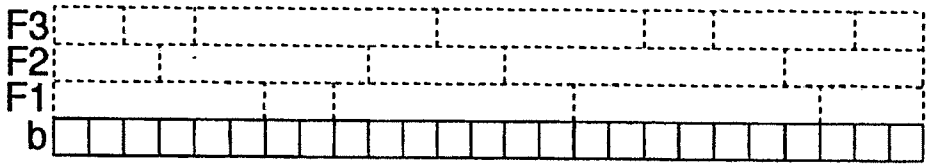


Figure 6

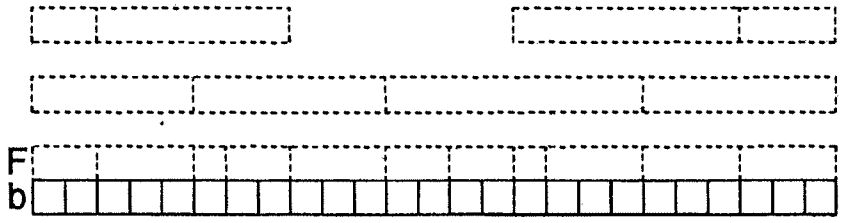


Figure 7

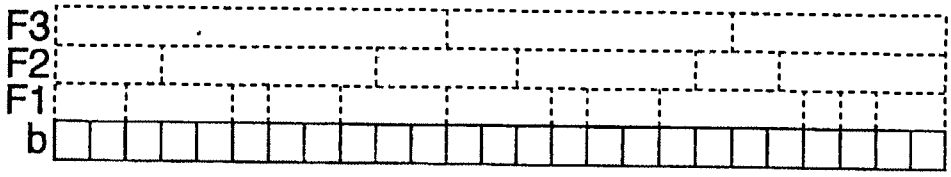


Figure 8

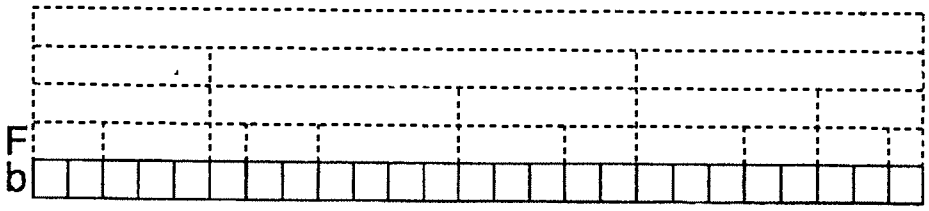


Figure 9

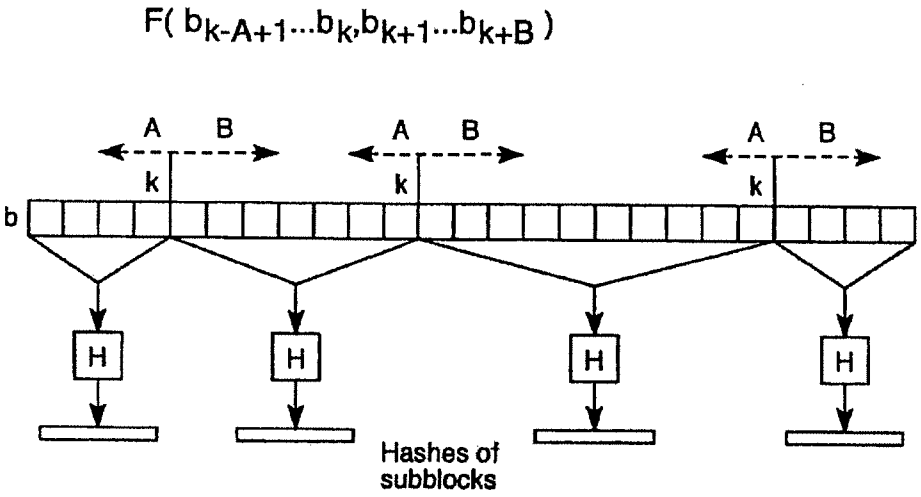


Figure 10

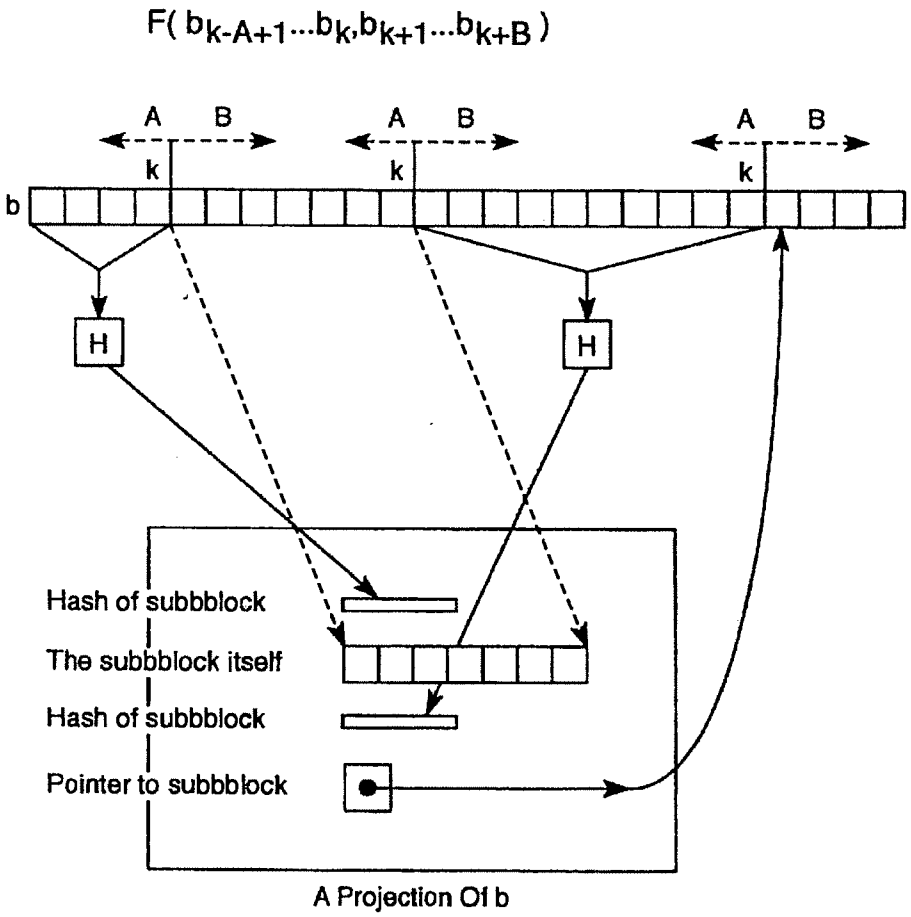

F($b_{k-A+1}...b_k,b_{k+1}...b_{k+B}$)
A
B
k
b
H
Hash of subbblock
The subbblock itself
Hash of subbblock
Pointer to subbblock
A Projection Of b

Figure 11

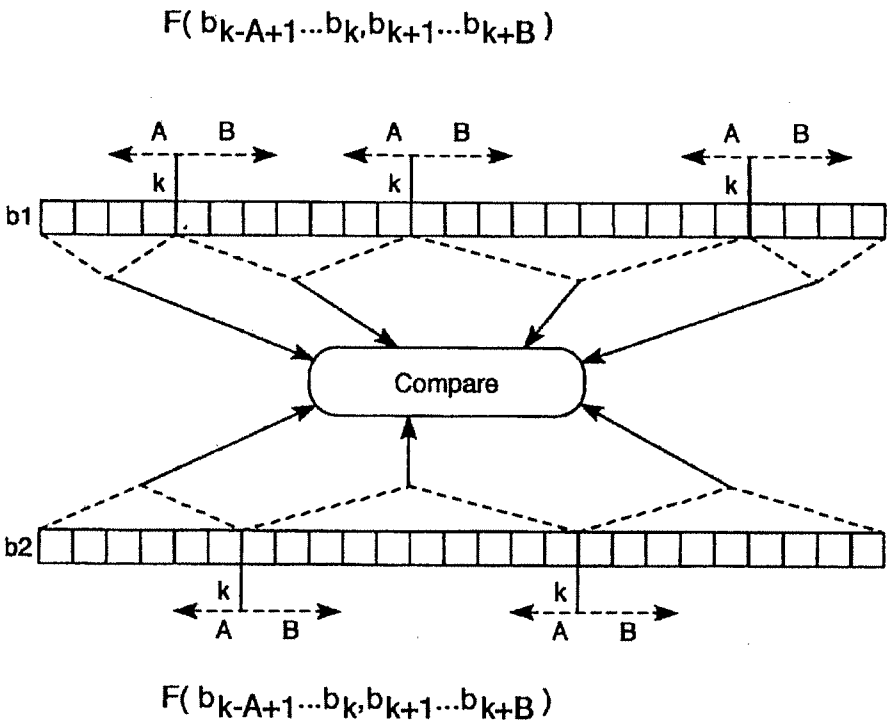


Figure 12

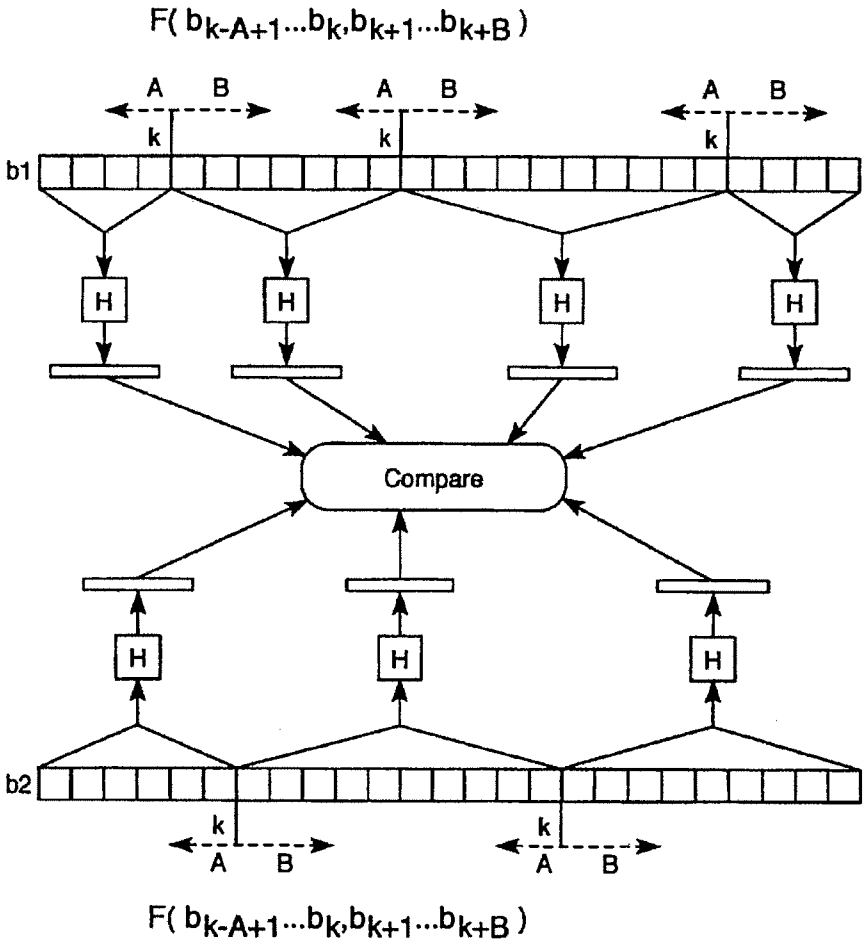


Figure 13

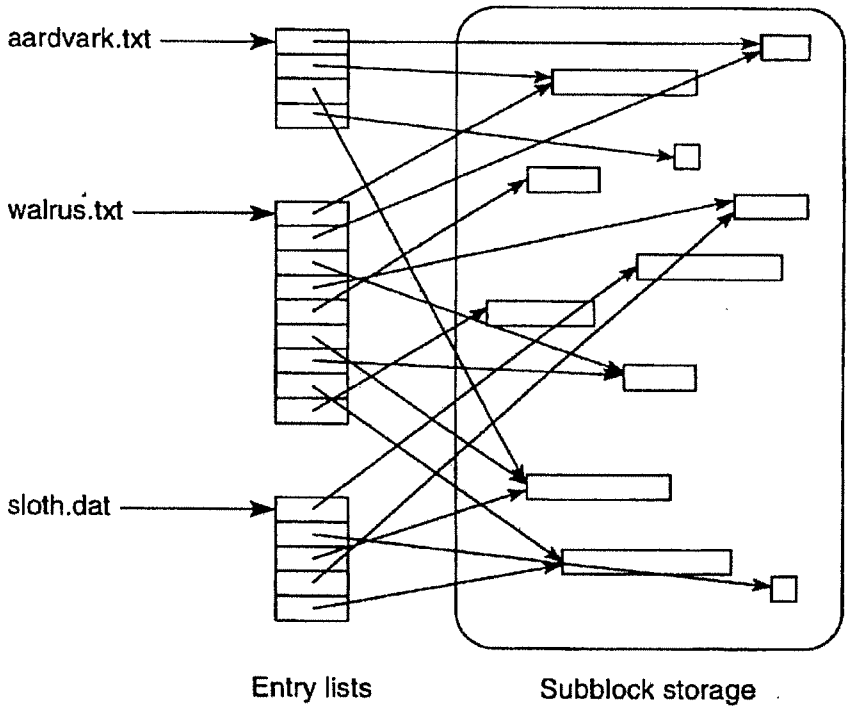

aardvark.txt
walrus.txt
sloth.dat
Entry lists
Subblock storage

Figure 14

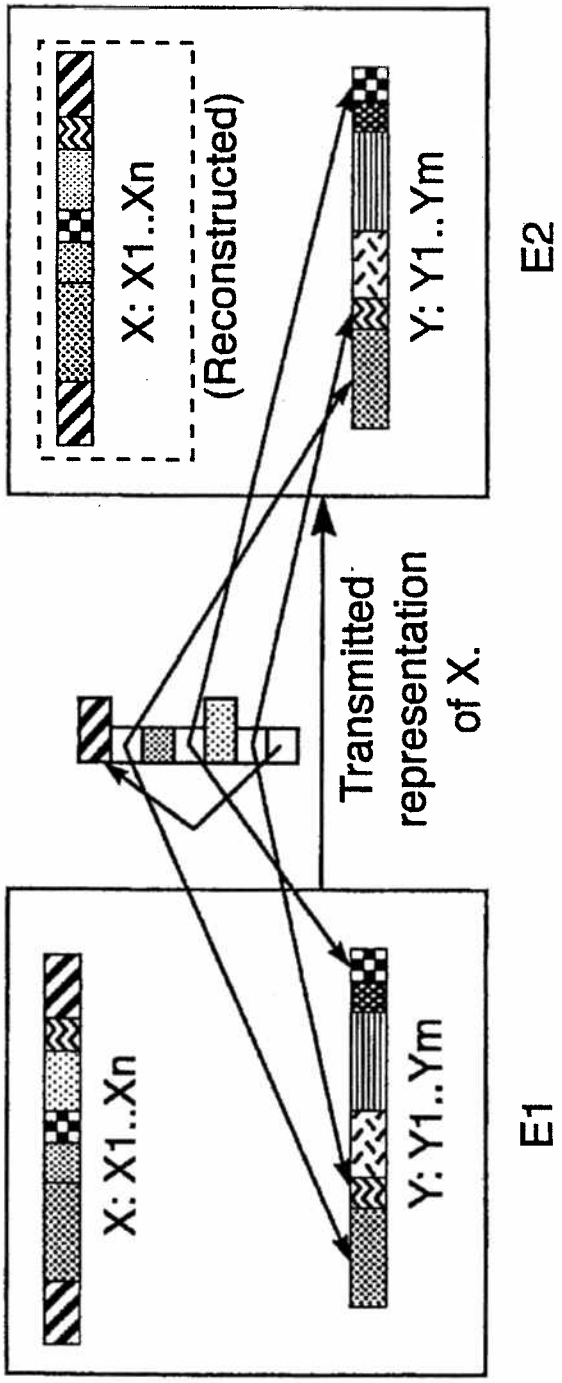

X: X1..Xn
(Reconstructed)
Y: Y1..Ym
E2
Transmitted representation of X.
X: X1..Xn
Y: Y1..Ym
E1

Figure 15

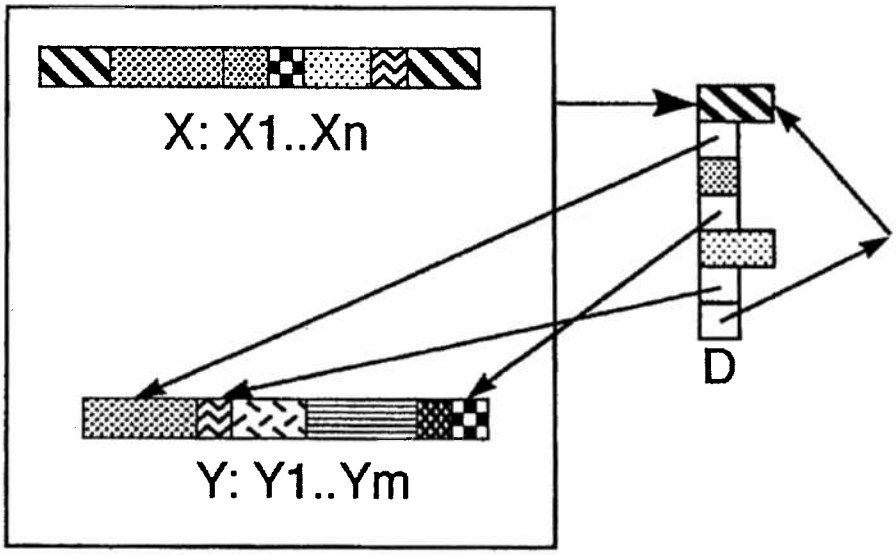


Figure 16

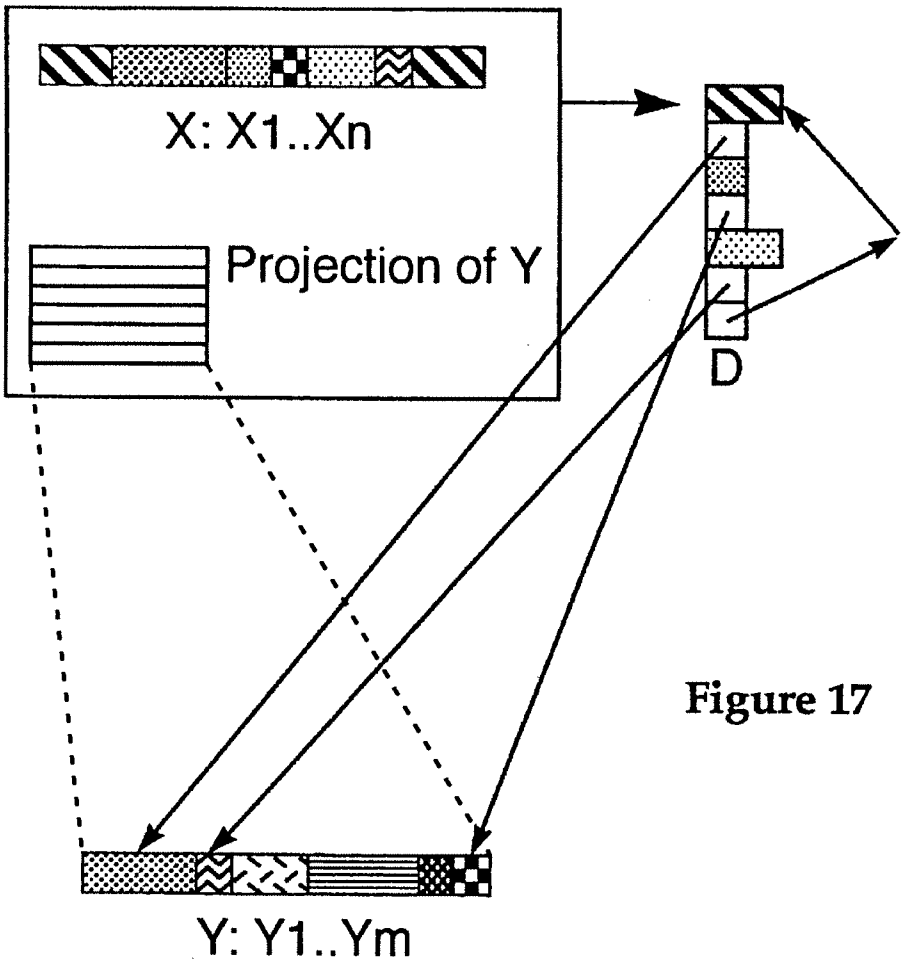


Figure 17

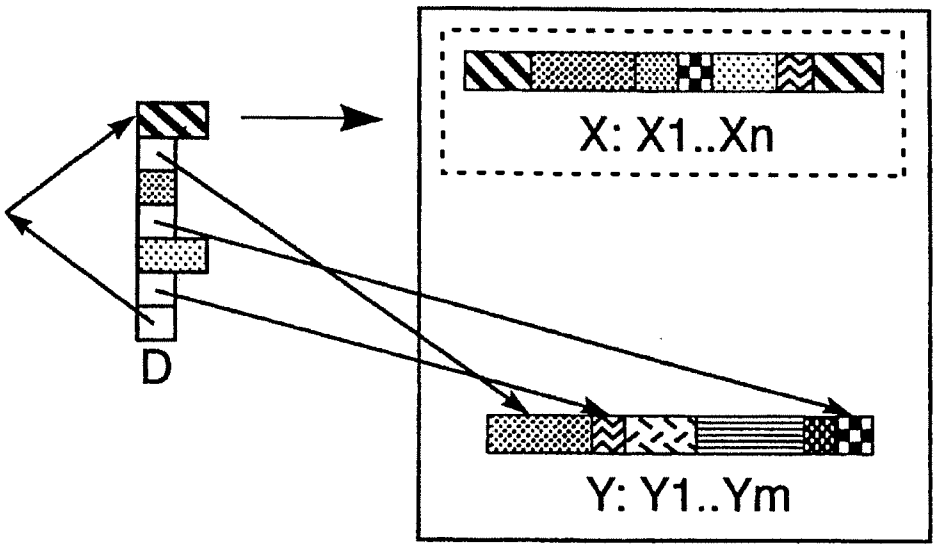


Figure 18

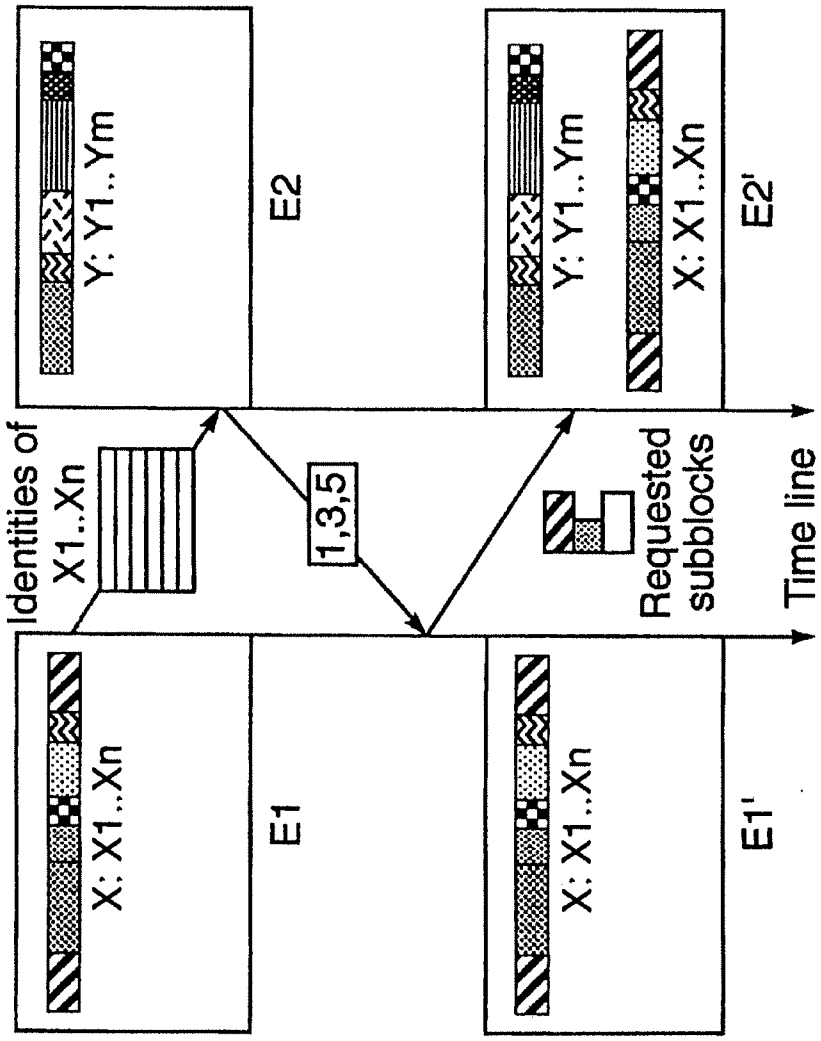

Y: Y1..Ym
E2
Y: Y1..Ym
X: X1..Xn
E2'
Identities of
X1..Xn
1,3,5
Requested
subblocks
Time line
X: X1..Xn
E1
X: X1..Xn
E1'

Figure 19

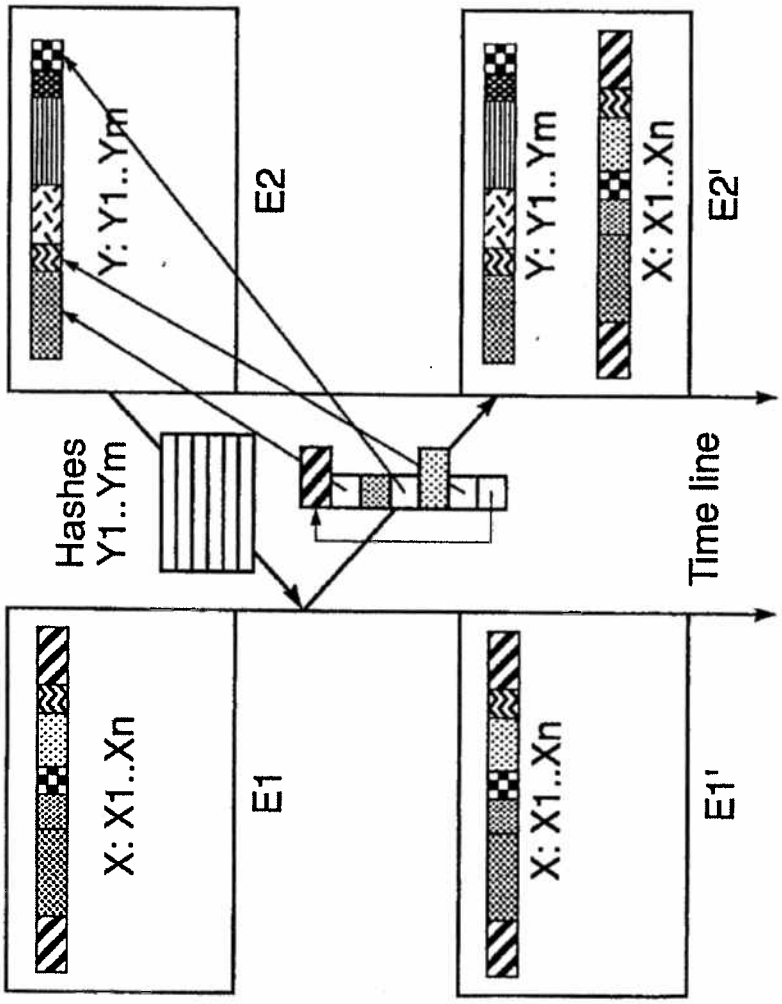

E2
Y: Y1..Ym
E2'
Y: Y1..Ym
X: X1..Xn
Hashes
Y1..Ym
Time line
E1
X: X1..Xn
E1'
X: X1..Xn

Figure 20

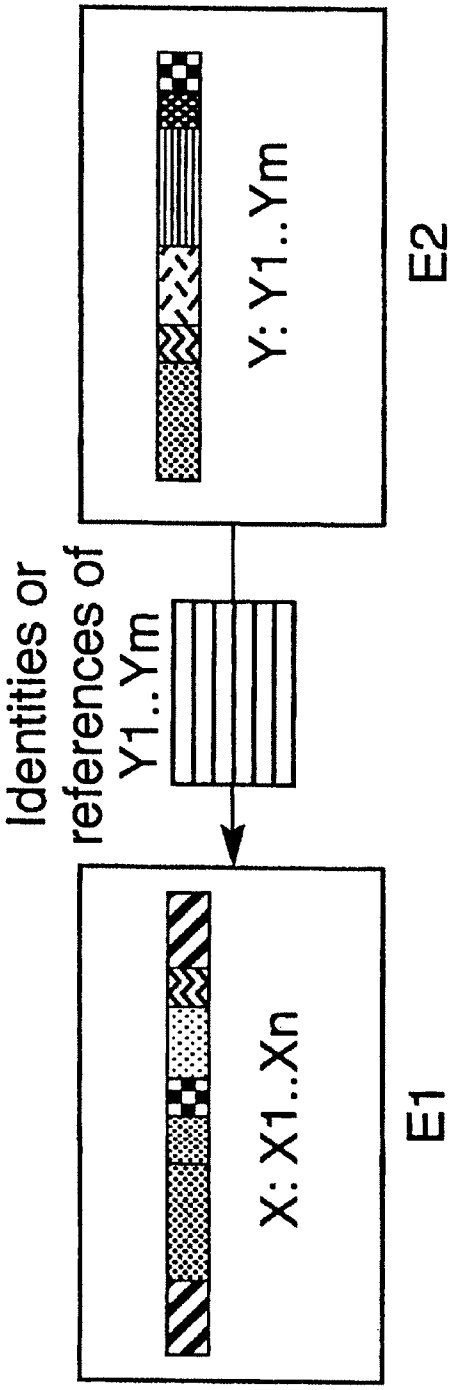


Figure 21

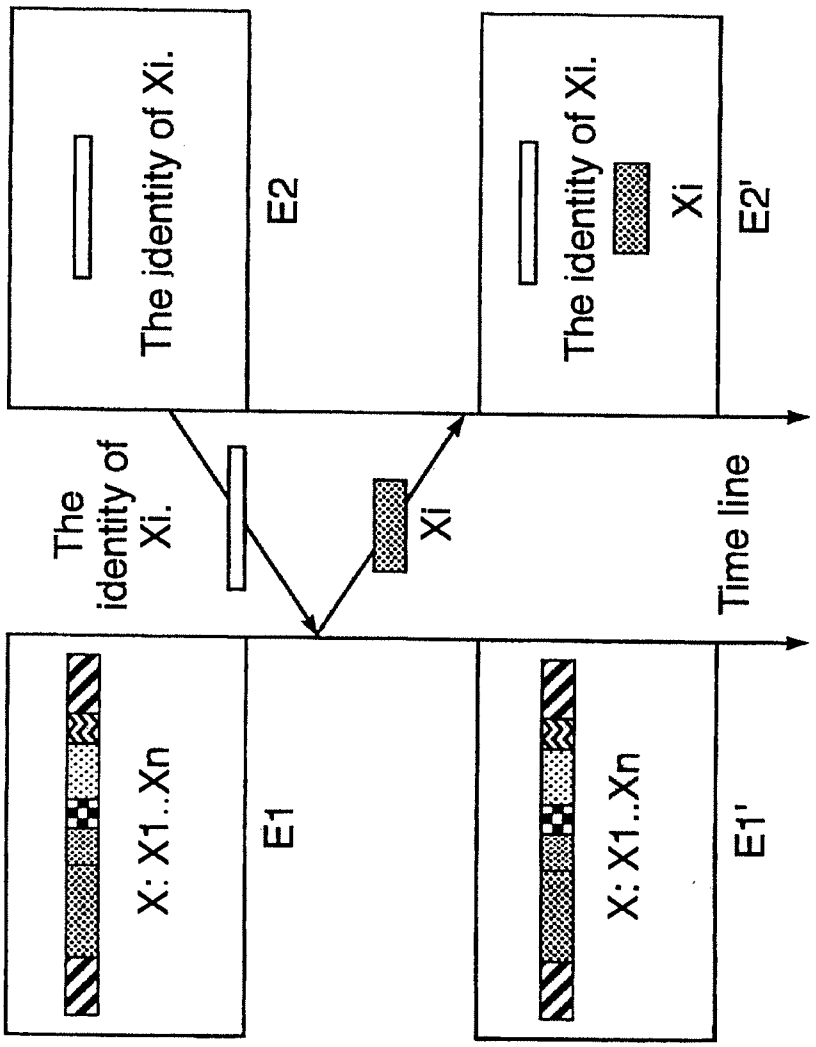

The identity of Xi.
E2
The identity of Xi.
Xi
E2'
The identity of Xi.
Xi
Time line
X: X1..Xn
E1
X: X1..Xn
E1'

Figure 22

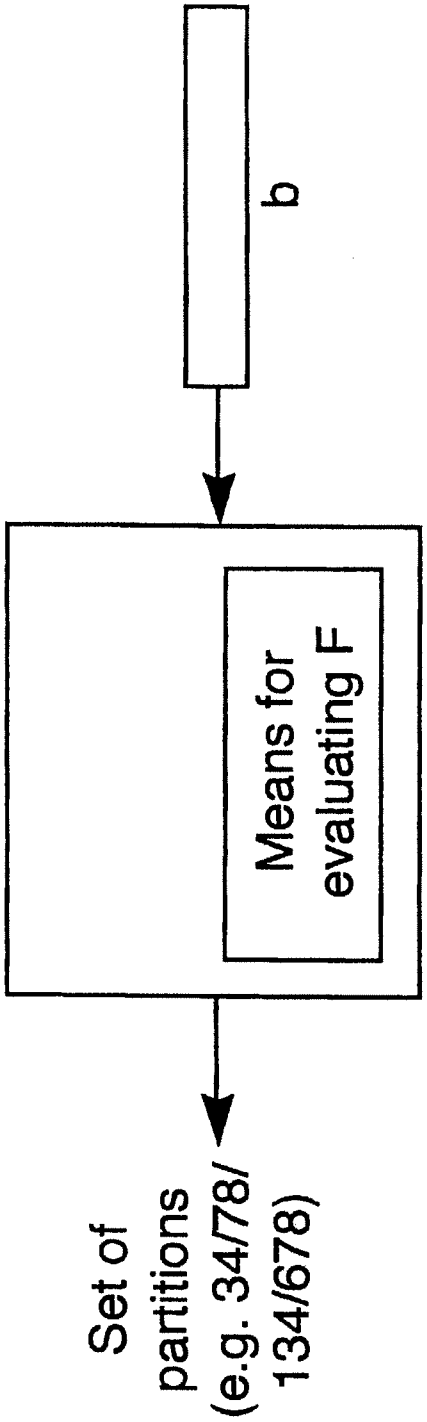

b
Means for evaluating F
Apparatus
Set of partitions (e.g. 34/78/ 134/678)

Figure 23

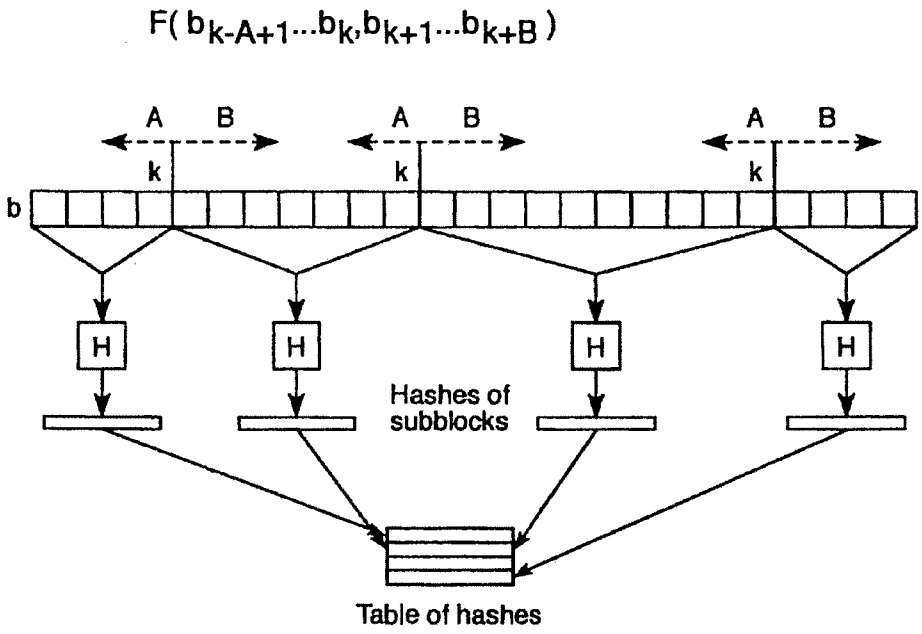


Figure 24

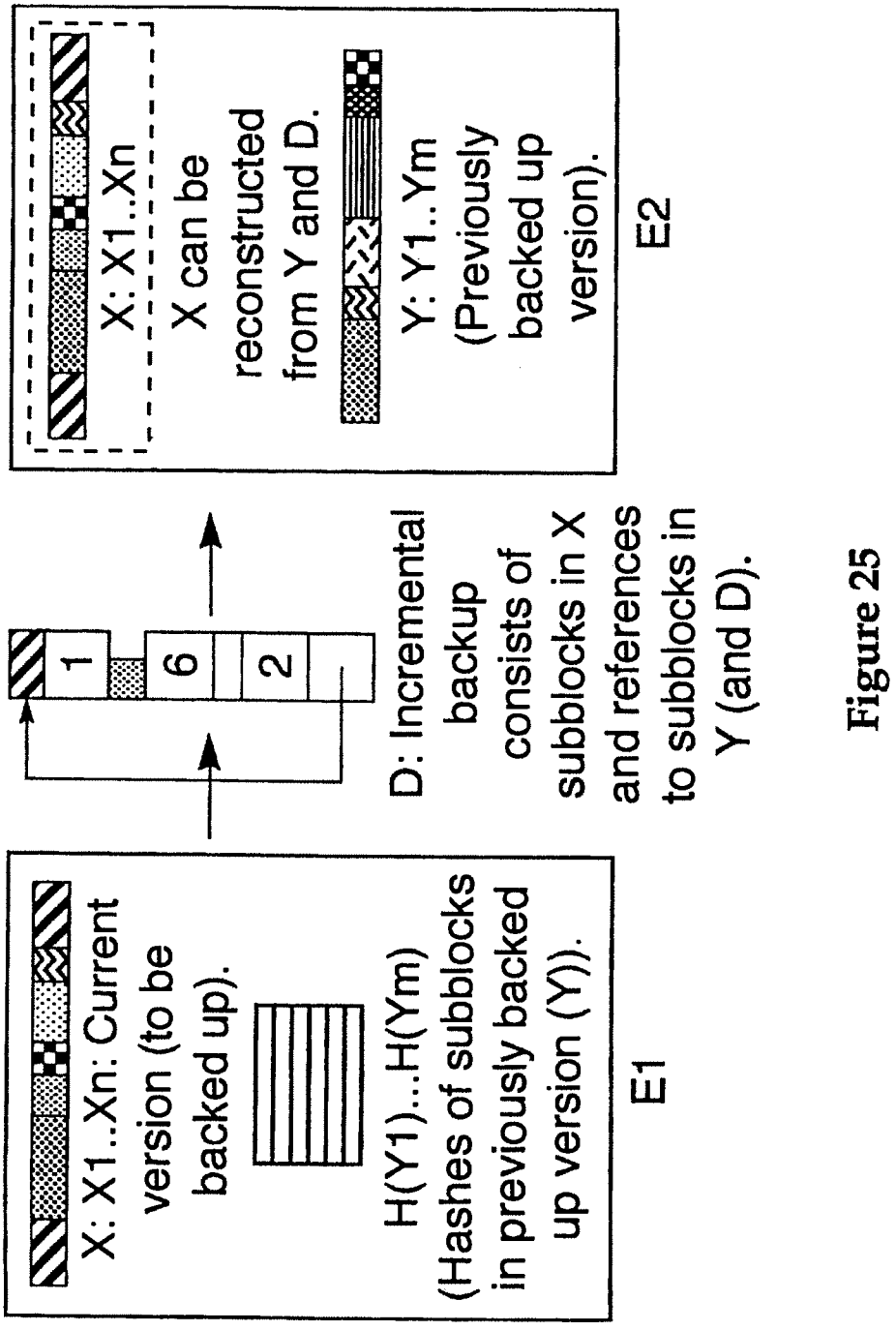

X: X1..Xn
X can be reconstructed from Y and D.
Y: Y1..Ym (Previously backed up version).
E2
1
6
2
D: Incremental backup consists of subblocks in X and references to subblocks in Y (and D).
X: X1..Xn: Current version (to be backed up).
H(Y1)...H(Ym) (Hashes of subblocks in previously backed up version (Y)).
E1

Figure 25

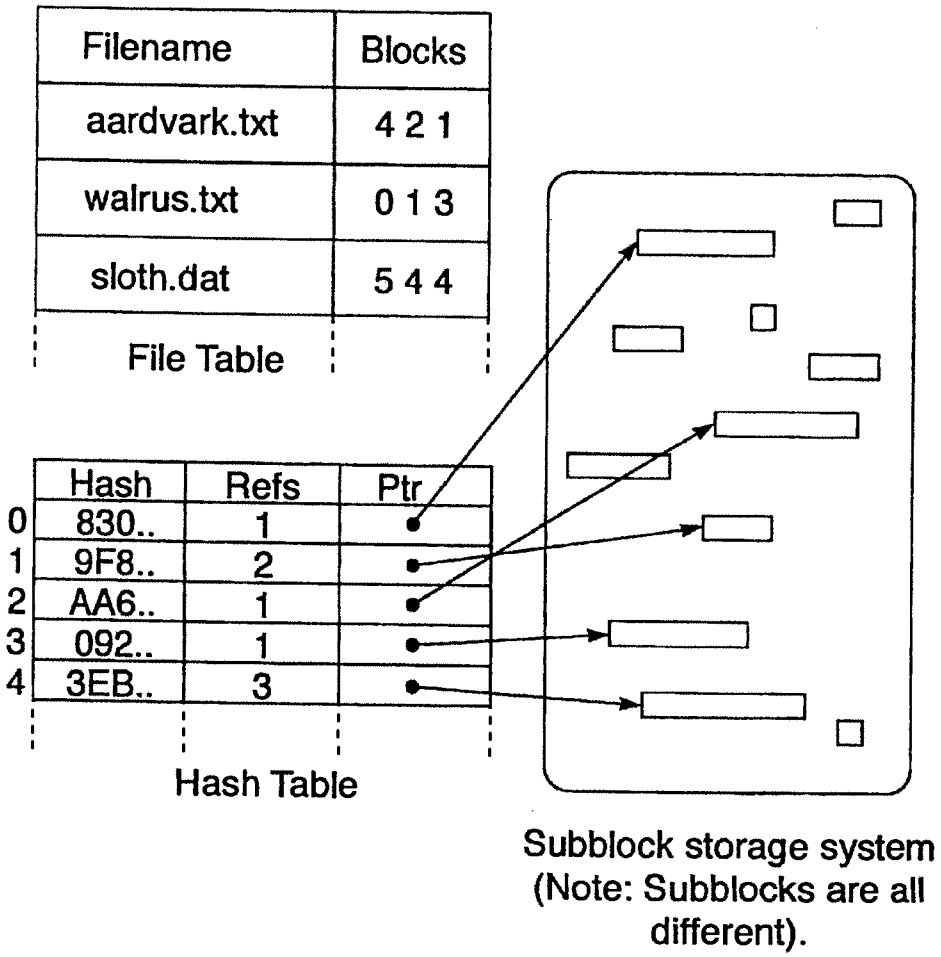

Filename
Blocks
aardvark.txt
4 2 1
walrus.txt
0 1 3
sloth.dat
5 4 4
File Table
Hash
Refs
Ptr
0
830..
1
1
9F8..
2
2
AA6..
1
3
092..
1
4
3EB..
3
Hash Table
Subblock storage system
(Note: Subblocks are all
different).

Figure 26

METHOD FOR PARITTIONING A BLOCK OF DATA INTO SUBBLOCKS AND FOR STORING AND COMMUNCATING SUCH SUBBLOCKS

INTRODUCTION

The present invention provides a method and apparatus for partitioning one or more blocks of data into subblocks for the purpose of communicating and storing such subblocks in an efficient manner.

BACKGROUND

Much of the voluminous amount of information stored, communicated, and manipulated by modern computer systems is duplicated within the same or a related computer system. It is commonplace, for example, for computers to store many slightly differing versions of the same document. It is also commonplace for data transmitted during a backup operation to be almost identical to the data transmitted during the previous backup operation. Computer networks also must repeatedly carry the same or similar data in accordance the requirements of their users.

Despite the obvious benefits that would flow from a reduction in the redundancy of communicated and stored data, few computer systems perform any such optimization. Some instances can be found at the application level, one example being the class of incremental backup utilities that save only those files that have changed since the most recent backup. However, even these utilities do not attempt to exploit the significant similarities between old and new versions of files, and between files sharing other close semantic ties. This kind of redundancy can be approached only by analysing the contents of the files.

The present invention addresses the potential for reducing redundancy by providing an efficient method for identifying identical portions of data within a group of blocks of data, and for using this identification to increase the efficiency of systems that store and communicate data.

SUMMARY OF THE INVENTION

To identify identical portions of data within a group of blocks of data, the blocks must be analysed. One simple approach is to divide the blocks into fixed-length (e.g. 512-byte) subblocks and compare these with each other so as to identify all identical subblocks. This knowledge can the be used to manage the blocks in more efficient ways.

Unfortunately, the partitioning of blocks into fixed-length subblocks does not always provide a suitable framework for the recognition of duplicated portions of data, as identical portions of data can occur in different sizes and places within a group of blocks of data. FIG. 1 shows how division into fixed-size subblocks of two blocks (whose only difference is the insertion of a single byte ('X')) fails to generate identical subblocks. A comparison of the two groups of subblocks would reveal no identical pairs of subblocks even thought the two original blocks differ by just one character.

A better approach is to partition each block using the data in the block itself to determine the position of the partitions.

In an aspect of the invention, the blocks are partitioned at boundaries determined by the content of the data itself. For example, a block could be partitioned at each point at which the preceding three bytes has to a particular constant value. FIG. 2 shows how such a data dependent partitioning could turn out, and contrasts it with a fixed-length partitioning. in FIG. 3 data independent partitioning generates seven distinct subblocks whereas the data-dependent partitioning generates just four, allowing much of the similarity between the two blocks to be detected.

The fact that a partitioning is data dependent does not imply that it must incorporate any knowledge of the syntax or semantics of the data. So long as the boundaries are positioned in a manner dependent on the local data content, identical subblocks are likely to be formed from identical portions of data, even if the two portions are not identically aligned relative to the start of their enclosing blocks (FIG. 3).

Once the group of blocks has blocks has been partitioned into subblocks, the resulting group of subblocks can be manipulated in a manner that exploits the occurrence of duplicate subblocks. This leads to a variety of applications, some of which are described below. However, the application of a further aspect of the invention leads to even greater benefits.

In a further aspect of the invention, the hash of one or more subblocks is calculated. The hash function can be an ordinary hash function or one providing cryptographic strength. The hash function maps each subblock into a small tractable value (e.g. 128 bits) that provides an identity of the subblock. These hashes can usually be manipulated more efficiently than their corresponding subblocks.

Some applications of aspects of this invention are:

Fine-grained incremental backups: Conventional incremental backup technology uses the file as the unit of backup. However, in practice many large files change only slightly, resulting in a wasteful re-transmission of changed files. By storing the hashes of subblocks of the previous versions of files, the transmission of unchanged subblocks can be eliminated.

Communications: By providing a framework for communicating the hashes of subblocks, the invention can eliminate the transmission of subblocks already possessed by the receiver.

Differences: The invention could be used as the basis of a program that determines the areas of similarity and difference between two blocks.

Low-redundancy file system: Data stored in a file system can be partitioned into subblocks whose hashes can be compared so as to eliminate the redundant storage of identical subblocks.

Virtual memory: Virtual memory could be organized by subblock using a table of hashes to determine if a subblock is somewhere in memory.

Clarification of Terms

The term block and subblock both refer, without limitation, to finite blocks or infinite blocks (sometimes called streams) of zero or more bits or bytes of digital data. Although the two different terms ("blocks" and "subblock") essentially describe the same substance (digital data), the two different terms have been employed in this specification to indicate the role that a particular piece of data is playing. The term "block" is usually used to refer to raw data to be manipulated by aspects of the invention. The term "subblock" is usually used to refer to a part of a block. "Blocks" are "partitioned" into "subblocks".

The term partition has its usual meaning of exhaustively dividing an entity into mutually exclusive parts. However, within this specification, the term also includes cases where:

Not all of the block is subdivided.

Multiple overlapping subblocks are formed.

A natural number is a non-negative integer (0, 1, 2, 3, 4, 5, . . .).

Where the phrase zero or more is used, this phrase is intended to encompass the degenerate case where the objects being enumerated are not considered at all, as well as the case where zero or more objects are used.

BRIEF DESCRIPTION

The following aspects of this invention are numbered for reference purposes. The terms "block" and "subblock" refer to blocks and subblocks of digital data.

1. In an aspect of the invention, the invention provides a method for organizing a block b of digital data for the purpose of storage, communication, or comparison, by partitioning said block into subblocks at one or more positions k|k+1 within said block for which b[k−A+1 . . . k+B] satisfies a predetermine constraint, where A and B are natural numbers.

Note: The specification of this aspect encompasses the degenerate case in which either A or B is zero. The specification also includes the case where the constraint does not pay attention to some parts of b[k−A+1 . . . k+B]. For example, a constraint that pays attention only to (say) b[k−3] and b[k+2] would fall under the classes of constraint corresponding to A≧4 and B≧2.

2. In a further aspect of the invention, the invention provides a method according to aspect 1 in which the constraint comprises the hash of some or all of b[k−A+1 . . . k+B].

3. In a further aspect of the invention, the invention provides a method according to aspect 1, for locating the nearest subblock boundary on a side of a position p|p+1 within a said block, comprising the step of:

a. Evaluating whether said predetermined constraint is satisfied at each position k|k+1, for increasing (or decreasing) k, where k starts with the value p.

4. In a further aspect of the invention, the invention provides a method according to aspect 1, wherein one or more bounds are imposed on the size of one or more subblocks.

5. In a further aspect of the invention, the invention provides a method according to aspect 1, wherein additional subblocks are formed from one or more groups of subblocks.

6. In a further aspect of the invention, the invention provides a method according to aspect 1, wherein an additional hierarchy of subblocks is formed from one or more groups of contiguous subblocks.

7. In a further aspect of the invention, the invention provides a method according to one of aspects 1 to 6, comprising the further step of:

b. Calculating the hash of each of one or more of said subblocks.

Note: The resulting collection of hashes is particularly useful if H is a strong one-way hash function.

8. In a further aspect of the invention, the invention provides a method according to one of aspects 1 to 6, comprising the further step of:

b. Forming a projection of said block, being an ordered or unordered collection of elements, wherein each element consists of a subblock, an identity of a subblock, or a reference of a subblock.

Note: The specification of this aspect is intended to admit collections that contain a mixture of various kinds of identities and references.

Note: In most applications, the output of this aspect will be an ordered list of hashes of the subblocks of the block.

9. In a further aspect of the invention, the invention provides a method for comparing one or more blocks, comprising the steps of:

a. Partitioning one or more of said blocks into one or more subblocks in accordance with one of aspects 1 to 6.

b. Forming a projection of each said block, being an ordered or unordered collection of elements, wherein each element consists of a subblock, an identity of a subblock, or a reference of a subblock.

c. Comparing the elements of said projections of said blocks.

10. In a further aspect of the invention, the invention provides a method for representing one or more blocks, comprising:

(i) A collection of subblocks;

(ii) Block representatives (e.g. filenames) which are mapped to lists of entries that ideotify subblocks;

whereby the modification of one of said blocks involves the following steps:

a. Partitioning some or all of said modified block into subblocks in accordance with one of aspects 1 to 6;

b. Adding to said collection of subblocks zero or more subblocks that are not already in said collection, and updating said subblock list associated with said modified block.

11. In a further aspect of the invention, the invention provides a method according to aspect 10, in which step b is replaced by:

b. Removing from said collection of subblocks zero or more subblocks, and updating said subblock list associated with said modified block.

12. In a further aspect of the invention, the invention provides a method according to aspect 10, in which step b is replaced by:

b. Adding to said collection of subblocks zero or more subblocks that are not already in the collection, removing from said collection of subblocks zero or more subblocks, and updating said subblock list associated with said modified block.

13. In a further aspect of the invention, the invention provides a method for an entity E1 to communicate a block X to E1 where E1 possesses the knowledge that E2 possesses a group Y of subblocks $Y_1 \ldots Y_m$, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with one of aspects 1 to 6;

b. Transmitting from E1 to E2 the contents of zero or more subblocks in X, and the remaining subblocks as references to subblocks in $Y_1 \ldots Y_m$ and to subblocks already transmitted.

Note: In most implementations of this aspect, the subblocks whose contents are transmitted will be those in X that are not in Y, and for which no identical subblock has previously been transmitted.

Note: To posses knowledge that E2 possesses $Y_1 \ldots Y_m$, E1 need not actually posses $Y_1 \ldots Y_m$ itself. E1 need only posses the identities of $Y_1 \ldots y_m$ (e.g. the hashes of each subblock $Y_1 \ldots Y_m$). This specification is intended to admit any other representation in which E1 may have the knowledge that E2 possesses (or has access to) $Y_1 \ldots Y_m$. In particular, the knowledge may take the form of a projection of Y.

Note: It is implicit in this aspect the E1 will be able to use comparison (or other methods) to use its knowledge of E2's possession of Y to determine the set of subblocks that are common to both X and Y. For example, if E1 possessed the hashes of the subblocks of Y, it could compare them to the hashes of the subblocks of X to determine the subblocks common to both X and Y. Subblocks that are not common can be transmitted explicitly. Subblocks that are common to both X and Y can be transmitted by transmitting a reference to the subblock.

14. In a further aspect of the invention, the invention provides a method for an entity E1 to communicate one or more subblocks of a group X of subblocks $X_1 \ldots X_n$ to E2 where E1 possesses the knowledge that E2 possesses the blocks Y, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with one or aspects 1 to 6;

b. Transmitting from E1 to E2 the contents of zero or more subblocks in X, and the remaining subblocks as references to subblocks in Y and to subblocks already transmitted.

15. In a further aspect of the invention, the invention provides a method for an entity E1 to communicate a block X to E2 where E1 possesses the knowledge that E2 possesses block Y, comprising the following steps:

a. Partitioning in accordance with one of aspects 1 to 6, X into subblocks $X_1 \ldots X_n$ and Y into subblocks $Y_1 \ldots Y_m$;

b. Transmitting from E1 to E2 the contents of subblocks in X, and the remaining subblocks as references to subblocks in Y and to subblocks already transmitted.

16. In a further aspect of the invention, the invention provides a method for constructing a block D from a block X and a group Y of subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with one of aspects 1 to 6;

b. Constructing D from one or more of the following: the contents of zero or more subblocks in X, references to zero or more subblocks in Y, and references to zero or more subblocks in D.

Note: Step b above is intended to encompass the case where a mixture of the elements it describes is used.

17. In a further aspect of the invention, the invention provides a method for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a block Y such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with one of aspects 1 to 6;

b. Constructing D from one or more of the following: the contents of zero or more subblocks in X, references to zero or more subblocks in Y, and references to zero or more subblocks in D.

18. In a further aspect of the invention, the invention provides a method for constructing a block D from a block X and a block Y such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning in accordance with one of aspects 1 to 6, X into subblocks $X_1 \ldots X_n$ and Y into subblocks $Y_1 \ldots Y_m$;

b. Constructing D from one or more of the following: the contents of zero or more subblocks in X, references to zero or more subblocks in Y, and references to zero or more subblocks in D.

19. In a further aspect of the invention, the invention provides a method for constructing a block D from a block X and a projection of Y, said projection comprising an ordered or unordered collection of elements wherein each element consists of a subblock in Y, an identity of a subblock in Y, or a reference of a subblock in Y, such that X can be constructed from Y and D, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with one of aspects 1 to 6;

b. Constructing D from one or more of the following: the contents of zero or more subblocks in X, references to zero or more subblocks in Y, and references to zero or more subblocks in D.

20. In a further aspect of the invention, the invention provides a method for constructing a block X from a block Y and a block D, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with one or aspects 1 to 6;

b. Constructing X from D and Y by constructing the subblocks of X based on one or more of:

(i) subblocks contained within D;

(iii) references in D to subblocks in Y;

(iii) references in D to subblocks in D;

21. In a further aspect of the invention, the invention provides a method for constructing a group X of subblocks $X_1 \ldots X_n$ from a block Y and a block D, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with one of aspects 1 to 6;

b. Constructing $X_1 \ldots X_n$ from D and Y based on one or more of:

(i) subblocks contained within D;

(iii) references in D to subblocks in Y;

(iii) references in D to subblocks in D;

22. In a further aspect of the invention, the invention provides a method for communicating a data block X from one entity E1 to another entity E2 comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with one of aspects 1 to 6;

b. Transmitting from E1 to E2 an identity of one or more subblocks;

c. Transmitting from E2 to E1 information communicating the presence or absence of subblocks at E2;

d. Transmitting from E1 to E2 at least the subblocks identified in step (c) as not being present at E2.

Note: The information communicated in step (c) could take the form of a bitmap (or a compressed bitmap) corresponding to the subblocks referred to in step (a). It could also take many other forms.

Note: If a group of subblocks are to be transmitted, the above steps could be performed completely for each subblock before moving onto the next subblock. The steps could be applied to any subgroup of subblocks.

23. In a further aspect of the invention, the invention provides a method for communicating a block X from one entity E1 to another entity E2, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with one of aspects 1 to 6;

b. Transmitting from E2 to E1 information communicating the presence or absence at E2 of members of a group Y or subblocks $Y_1 \ldots Y_m$;

c. Transmitting from E1 to E2 the contents of zero or more subblocks in X, and the remaining subblocks as references to subblocks in $Y_1 \ldots Y_m$ and to subblocks transmitted.

24. In a further aspect of the invention, the invention provides a method for an entity E2 to communicate to an entity E1 the fact that E2 possesses a block Y, comprising the following steps:

a. Partitioning Y into subblocks $Y_1 \ldots Y_m$ in accordance with one or aspects 1 to 6;

b. Transmitting from E2 to E1 references of the subblocks $Y_1 \ldots Y_m$.

25. In a further aspect of the invention, the invention provides a method for an entity E1 to communicate a subblock X_i to an entity E2, comprising the following steps:

a. Partitioning X into subblocks $X_1 \ldots X_n$ in accordance with one of aspects 1 to 6;

b. Transmitting from E2 to E1 an identity of X_i;

c. Transmitting X_i from E1 to E2.

Note: This aspect applies (among other applications) to the case of a network server E1 that serves subblocks to clients such as E2, given the identities (e.g. hashes) of the requested subblocks.

26. In a further aspect of the invention, the invention provides a method according to one of aspects 1 to 6, wherein said subblocks are compared by comparing the hashes of said subblocks.

27. In a further aspect of the invention, the invention provides a method according to one of aspects 1 to 6, wherein subsets of identical subblocks within a group of one or more subblocks are found, by inserting each subblock, an identity of each subblock, a reference of each subblock, or a hash of each subblock, into a data structure.

28. In further aspect of the invention, the invention provides an apparatus for organizing a block b of digital data for the purpose of storage, communication, or comparison, by partitioning said block into subblocks at one or more positions k|k+1 within said block for which b[k−A+1 . . . k+B] satisfies a predetermined constraint, where A and B are natural numbers.

Note: The specification of this aspect encompasses the degenerate case in which either A or B is zero. The specification also includes the case where the constraint does not pay attention to some parts of b[k−A+1 . . . k+B]. For example, a constraint that pays attention only to (say) b[k−3] and b[k+2] would fall under the classes of constraint corresponding to A≧4 and B≧2.

29. In a further aspect of the invention, the invention provides an apparatus according to aspect 28 in which the constraint comprises the hash of some or all of b[k−A+1 . . . k+B].

30. In a further aspect of the invention, the invention provides an apparatus according to aspect 28, for locating the nearest subblock boundary on a side of a position p|p+1 within a said block, comprising the step of:

a. Evaluating whether said predetermined constraint is satisfied at each position k|k+1, for increasing (or decreasing) k, where k starts with the value p.

BRIEF DESCRIPTION OF FIGURES

FIG. 1 shows how data can become "misaligned" relative to its containing blocks when data is inserted.

FIG. 2 shows how data can be divided into fixed-width subblocks or variable-width subblocks.

FIG. 3 shows how data-dependent partition move with the data when the data is shifted (e.g. by an insertion) H.

FIG. 4 depicts the data-dependent partitioning of a block b of data into subblocks using a constraint F.

FIG. 5 depicts the search within a block b for a subblock boundary using a constraint F.

FIG. 6 shows how a block may be subdivided in different ways using different partitioning constraints.

FIG. 7 shows how "higher order" subblocks can be constructed from one or more initial subblocks.

FIG. 8 shows how different partitioning functions can produce subblocks of differing average sizes.

FIG. 9 shows how subblocks can be organized into a hierarchy. Such a hierarchy can be constructed by progressively restricting a constraint F.

FIG. 10 depicts a method (and apparatus) for the partitioning of a block b into subblocks using a constraint F, and the calculation of the hashes of the subblocks using hash function H.

FIG. 11 depicts the partitioning of a block b into subblocks using a constraint F, and the projection of those subblocks into a structure consisting of subblock hashes, subblock data, and subblock references.

FIG. 12 depicts a method (and apparatus) for partitioning two blocks b1 and b2 into subblocks, using a constraint F, and then comparing the subblocks.

FIG. 13 depicts a method (and apparatus) for the partitioning using a constraint F, of two blocks b1 and b2 into subblocks, the calculation of the hashes of the subblocks using H, and the comparison of those hashes with each other to determine (among other things) subblocks common to both b1 and b2.

FIG. 14 depicts a method (and apparatus) for a file system that employs an aspect of the invention to eliminate the multiple storage of subblocks common to more than one file (or to different parts of the same file).

FIG. 15 depicts a method (and apparatus) for the communication of a block X from E1 to E2 where both E1 and E2 possess Y.

FIG. 16 depicts a method (and apparatus) for the construction of a block D from which X may be later reconstructed, given Y.

FIG. 17 depicts s method (and apparatus) for the construction of a block D from which X may be later reconstructed, given Y. In this case, the entity constructing D does not have access to Y, only to a projection of Y (being perhaps the hashes of the subblocks of Y).

FIG. 18 depicts a method (and apparatus) for the reconstruction of X from the blocks Y and D.

FIG. 19 depicts a method (and apparatus (E1 and E2 at each time)) for the communication of a block X from entity E2 where E2 already possesses Y.

FIG. 20 depicts a method (and apparatus (E1 and E2 at each time)) for the communication of a block X from entity E1 to entity E2 where E2 already possesses Y and where E2 first discloses to E1 information about Y.

FIG. 21 depicts a method (and apparatus) for the communication, from entity E2 to entity E1, information about a block (or group of subblocks) Y at E2.

FIG. 22 depicts a method (and apparatus (E1 and E2 at each time)) for the communication from entity E1 to entity E2 of subblock X_i following a request by entity E2 for the subblock X_i.

FIG. 23 depicts an apparatus for partitioning a block b (the input) using a constraint F. The output is a set of subblock boundary positions.

FIG. 24 depicts a method (and apparatus) for the partitioning of a block b into subblocks using constraint F, and the projection of those subblocks into a table of subblock hashes.

FIG. 25 depicts a method (and apparatus) for the transmission from entity E1 to E2 of a block X where E2 possesses Y and E1 possesses a table of the hashes of the subblocks of Y (a projection of Y).

FIG. 26 depicts a method (and apparatus) for a file system that employs an aspect of the invention to eliminate the multiple storage of data common to more than file (or to different parts of the same file).

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

This section contains a detailed discussion of mechanisms that could be used to implement aspects of the invention. It also contains examples of implementations of selected aspects of the invention. However, nothing in this section should be interpreted as a limitation on the scope of this patent.

Units of Information

Aspects of this invention can be applied at various levels of granularity of data. For example, if the data was treated

as a stream of bits, boundaries could be placed between any two bits. However, if the data was treated as a stream of bytes, boundaries would usually be positioned only between bytes. The invention could be applied with any unit of data, and in this document references to bits and bytes should usually be interpreted as admitting any granularity.

The Concept of Entity

At various places, this patent specification uses the term "entity" to describe an agent. This term is purposefully vague and is intended to cover all forms of agent including, but not limited to:

Computer systems.

Networks of computer systems.

Processes in computer systems.

File systems.

Components of software.

Dedicated computer systems.

Communications systems.

The Concepts of Identity and Reference

This patent specification frequently refers to "identities" of subblocks and "references" to subblocks. These terms are not intended to be defined precisely.

The identity of a subblock means any piece of information that could be used in place of the subblock for the purpose of comparison for identicality. Identities include, but are not limited to:

The subblock itself.

A hash of the subblock.

The subblock acts as its own identity because subblocks themselves can be compared with each other. Hashes of subblocks also act as identities of subblocks because hashes of subblocks can be compared with each other to determine if their corresponding subblocks are identical.

A reference to a subblock means any piece of information that could be used in practice by one entity to identify to another entity (or itself) a particularly valued subblock, where the two entities may already share some knowledge. For example, the two entities might each possess the knowledge that the other entity already possesses ten subblocks of known values having particular index values numbered one to ten.

Once two entities have a basis of shared knowledge, it is possible for them to identify a subblock in ways more concise than the transmission of an identity. A reference to a particularly valued subblock can take (without limitation) the following forms:

An identity.

An identifying number of a subblocks possessed by the receiver.

An identifying number of a subblock previously transmitted between the two communicants.

The location of the subblock in some shared data space.

As relative subblocks number.

Ranges of the above.

The concept of knowledge of a subblock is related to the concepts of identity and reference. An entity may have knowledge of a subblock (or knowledge that another entity possesses a subblock) without actually possessing the subblock itself. For example, it might possess an identity of the subblock or a reference to the subblock.

The Use of Ranges

In any situation where a group of values that have contiguous values (e.g. 6, 7, 8, 9) is to be communicated or stored, such a group can be represented using a range (e.g. 6–9) which may take up less communication time or storage space. Ranges can be applied to all kinds of things, such as index values and subblock numbers. In particular, if an entity notices that the references (to subblocks) that it is about to transmit are contiguous, it can replace the references with a range.

Ranges can be represented in any way that identifies the first and last element of the range. Three common representations are:

The first and last element of the range.

The first element and the length of the range.

The last element and the length of the range.

The concept of range can be generalized to include the compression of any group of values that exhibit compressible structure.

The Use of Backward References

References can be used not only to refer to data shared by two communicants at the start of a transmission, but can also be used to refer to data communicated at some previous time during the transmission.

For example, if an entity A notices that the subblock it is about to transmit to another entity B was not possessed by B at the start of the transmission, but has since been transmitted from A to B, then A could code the second instance of the subblock as a reference to the previous instance of the subblock. The range mechanism can be used here too.

No Requirement for Subblock Framing Information

It is possible that an entity E1 could transmit a group X of subblocks $X_1 \ldots X_n$ as a group to an entity E2 simply by sending the concatenation of the subblocks. There may be no need for any framing information (e.g. information at the start of each subblock giving the length of the subblock or "escape" codes to indicate subblock boundaries), as E2 is capable of partitioning X into $X_1 \ldots X_n$ itself.

No Requirement for Ordering Subblocks

If two entities E1 and E2 both possess the same unordered group Y of subblocks (or knowledge of such a group of subblocks) then even though E1 and E2 may not possess the subblocks in the same order, the subblocks can still be referred to using a subblock index or serial number. This is achieved by having E1 and E2 each sort their subblocks in accordance with some mutually agreed (or universally defined) ordering method and then number the subblocks in the resultant ordered group of subblocks. These number (or ranges of such numbers) can then be used to refer to the subblocks.

An Overview of Hash Functions

Although the use of a hash function is not essential in all aspects of this invention, hash functions provide such advantages in the implementation of this invention that an overview of them is warranted.

A hash function accepts a variable-length input block of bits and generates an output block of bits that is based on the input block. Most hash functions guarantee that the output

block will be of a particular length (e.g. 16 bits) and aspire to provide a random, but deterministic, mapping between the infinite set of input blocks and the finite set of output blocks. The property of randomness enables these outputs, called "hashes", to act as easily manipulated representatives of the original block.

Hash functions come in at least four classes of strength.

Narrow hash functions: Narrow hash functions are the weakest class of hash functions and generate output values that are so narrow (e.g. 16 bits) that the entire space of output values can be searched in a reasonable amount of time. For example, an 8-bit hash function would map any data block to a hash in the range 0 to 155. A 16-bit has function would map to a hash in the range 0 to 65535. Given a particular hash value, it would be possible to find a correspoodiog block simply by generating random blocks and feeding them into the narrow hash function until the searched-for value appeared. Narrow hash functions are usually used to arbitrarily (but deterministically) classify a set of data values into a small number of groups. As such, they are useful for constructing hash table data structures, and for detecting errors in data transmitted over noisy communication channels. Examples of this class: CRC-16, CRC-32. Fletcher checksum, the IP checksum.

Wide hash functions: Wide hash functions are similar to narrow hash functions except that their output values are significantly wider. At a certain point this quantitative difference implies a qualitative difference. In a wide hash function, the output value is so wide (e.g. 128 bits) that the probability of any two randomly chosen blocks having the same hashed value is negligible (e.g. about one in 10^{38}). This property enables these wide hashes to be used as "identities" of the blocks of data from which they are calculated. For example, if entity E1 has a block of data and sends the wide hash of the block to an entity E2, then if entity E2 has a block that has the same hash, then the a-priori probability of the blocks actually being different is negligible. The only catch is that wide hash functions are not designed to be non-invertible. Thus, while the space of (say)2^{128} values is too large to search in the manner described for narrow hash functions, it may be easy to analyse the hash function and calculate a block corresponding to a particular hash. Accordingly, E1 could fool E2 into thinking E1 had one block when it really had a different block. Examples of this class: any 128-bit CRC algorithm.

Weak one-way hash functions: Weak one-way hash functions are not only wide enough to provide "identity", but they also provide cryptographic assurance that it will be extremely difficult, given a particular hash value, to find a block corresponding to that hash value. Examples of this class: a 64-bit DES hash.

Strong one-way has functions: Strong one-way hash functions are the same as weak one-way hash functions except that they have the additional property of providing cryptographic assurance that it is difficult to find any two different blocks that have the same hash value, where the hash value is unspecified. Examples of this class: MD4, MD5, and SHA-1.

These four classes of hash provide a range of hashing strengths from which to choose. As might be expected, the speed of a hash function decreases with strength, providing a tradeoff, and different strengths are appropriate in different applications. However, the difference is small enough to admit the use of strong one-way hash functions in all bnt the most time-critical applications.

The term cryptograpic hash is often used to refer to hashes that provide cryptographic strength, encompassing both the class of weak one-way hash functions and the class of strong one-way hash functions. However, as strong one-way hash functions are almost preferable to weak one-way hash functions, the term "cryptographic hash" is used mainly to refer to the class of strong one-way hash functions.

The present invention can employ hash functions in at least two roles:

1. To determine subblock boundaries.
2. To generate subblock identities.

Depending on the application, hash functions from any of the four classes above could be employed in either role. However, as the determination of subblock boundaries does not require identity or cryptographic strength, it would be inefficient to use hash functions from any bnt the weakest class. Similarly, the need for identity, the ever-present threat of subversion, and the minor performance penalty for strong one-way hash functions (compared to weak ones) suggests that nothing less than strong one-way hash functions should be used to calculate subblock identities.

The security dangers inherent in employing anything less than a strong one-way hash function to generate identities can be illustrated by considering a communications system or file system that incorporates the invention using any such weaker hash function. In such a system, an intruder could modify a subblock (to be manipulated by a target system) in such a way that the modified subblock has the same a hash as another subblock known by the intruder to be already present in the target system. This could result in the target system retaining its existing subblock rather than replacing it by a new one. Such a weakness could be used (for example) to prevent a target system from properly applying a security patch retrieved over a network.

Thus, while wide has functions could be safely used to calculate subblocks in systems not exposed to hostile humans, even weak one-way hash functions are likely to be insecure in those systems that are.

We now turn to the ways in which hashes of blocks or subblocks can actually be used.

The Use of Cryptographic Hashes

The theoretical properties of cryptographic hashes (and here is meant strong one-way hash functions) yield particularly interesting practical properties. Because such hashes are significantly wide, the probability of two randomly-chosen subblocks having the same hash is practically zero (for a 128-bit hash, it is about one in 10^{38}), and because it is computationally infeasible to find two subblocks having the same hash, it is practically guaranteed that no intelligent agent will be able to do so. The implication of these properties is that from a practical perspective, the finite set of hash values for a particular cryptographic hash algorithm is one-to-one with the infinite set of finite variable length subblocks. This theoretically impossible property manifests itself in practice because of the practical infeasibility of finding two subblocks that hash to the same value.

This property means that, for the purposes of comparison (for identically), cryptographic hashes may safely be used in place of the subblocks from which they were calculated. As most cryptographic hashes are only about 128 bits long, hashes provide an extremely efficient way to compare subblocks without requiring the direct comparison of the content of the subblocks themselves. Such comparisons can be used to eliminate many transmissions of information. For example, a subblock X_1 on a computer C1 in Sydney could be compared with a subblock Y_1 on a computer C2 in Boston

by a computer C3 in Paris, with the total theoretical network traffic being just 256 bits (C1 and C2 each send the 128-bit hash of their respective subblocks to C3 for comparison, and C3 compares the two hashes).

Some of the ways in which cryptographic hashes could be used in aspects of this invention are:

Cryptographic hashes can be used to compare two subblocks without having to compare, or requiring access to, the content of the subblocks.

If it is necessary to be able to determine whether a subblock T is identical to one of a group of subblocks, the subblocks themselves need not be stored, just a collection of their hashes. The hash of any candidate subblock can then be compared with the hashes in the collection to establish whether the subblock is in the group of subblocks from which the collection of hashes was generated.

Cryptographic hashes can be used to ensure that the partitioning of a block into subblocks and the subsequent reassembly of the subblocks into a reconstructed block is error-free. This can be done by comparing the hash of the original block with the hash of the reconstructed block.

If an entity E1 calculates the hash of a subblock X_1 and transmits it to E2, then if E2 possesses X_1, or even just the hash of X_1, then E2 can determine without any practical doubt that E1 possesses X1.

If an entity E1 passes a key (consisting of a block of bits) chosen at random to an entity E2, E2 may then prove to E1 that it possesses a subblock by sending E1 the hash of the concatenation of the key and the subblock. This mechanism could be used as an additional check in security applications.

If a group of subblocks must be compared so as to find all subsets of identical subblocks, the corresponding set of hashes of the subblocks may be calculated and compared instead.

Many of the uses of cryptographics hashes for subblocks can also be applied to blocks. For example, cryptographic hashes can be used to determine whether a block has changed at all since it was last backed up. Such a check could eliminate the need for further analysis.

Use of Hashes as a Safety Net

A potential disadvantage of deploying aspects of this invention is that it will add extra complexity to the systems into which it is incorporated. This increased complexity carries the potential to increase the chance of undetected failures.

The main mechanism of complexity introduced by many aspects of the invention is the partitioning of blocks (e.g. files) into subblocks, and the subsequent re-assembly of such subblocks. By partitioning a block into subblocks, a system creates the potential for subblocks to be erroneously added, deleted, rearranged, substituted, duplicated, or in some other way exposed to a greater risk of accidential error.

This risk can be reduced or eliminated by calculating the hash (preferably a cryptographic hash of the block before it is partitioned into subblocks, storing the hash with an entity associated with the block as a whole, and then later comparing the stored hash with a compound hash of the reconstructed block. Such a check would provide a very strong safety net that would virtually eliminate the risk of undetected errors arising from the use of this invention.

Choosing a Partitioning Constraint Function

Although the requirements for the block partitioning constraint (e.g. in the form of a constraint function F) are not stringent, care should be taken to select a function that suits the application to which it is to be applied.

In situation where the data is highly structured and knowledge of the data is available, a choice of an F that tends to place subblock boundaries at positions in the data that correspond to obvious boundaries in the data could be advantageous. However, in general, F should be chosen from the class of narrow hash functions. Use of a narrow hash function for F provides both efficiency and a (deterministic) randomness that will enable the implementation to operate effectively over a wide-range of data.

One of the most important properties of F is the probability that F will place a boundary at any particular point when applied to completely random data. For example, a function with a probability of one would produce a boundary between each bit (or byte), whereas a function with a probability of zero would never produce any boundaries at all. In a real application, a more moderate probability would be chosen (e.g. 1/1024) so as to yield useful subblock sizes. The probability can be tuned to suit the application.

We end this section with an example of a narrow hash function that has been implemented and tested and seems to perform well on a variety of data types. The hash function calculates a hash value from three bytes.

$$H(b_1, b_2, b_3)=((40543\times((b_1<<8) \otimes (b_2<<4) \otimes b_3))>>4) \mid \rho$$

The following notation has been used. "×" is multiplication. "<<" is left bit shift. ">>" is right bit shift. "⊗" is exclusive or. "|" is modulo. The constant ρ is the inverse of the probability of placing a boundary at an arbitrary position in a randomly generated block of data, and can be set to any integer value in [0,65535]. However, in practice it seems to be advantageous to choose values that are prime (Mersenne primes seem to work well). The value 40543 was chosen carefully in accordance with the hash function design guidelines provided in pages 508–613 of the book:

Knuth D. E., "The Art of Computer Programming: Volume 3: Sorting and Searching", Addision Wesley, 1973.

The function generates a value in the range [0, ρ−1] and can be used in practice by placing a boundary at each point where the preceding three bytes hash to a predetermined constant value V. This would imply that its arguments $b_1 \ldots b_3$ correspond to the argument A in aspect one above. To avoid pathological behaviour in the commonly occurring case of runs of zeros, it is wise to choose a non-zero value for V.

In a real implementation, ρ was set to 511 and V was set to one.

Placing an Upper and Lower Bound on the Subblock Size

The use of data-dependent subblock boundaries provides a way to deterministically partition similar portions of data in a context-independent way. However, if artifical bounds are not placed on the subblock size, particular kinds of data will yield subblocks that are too large or too small to be effective. For example, if a file contains a block of a million identical bytes, any deterministic constraint (that operates at the byte level) must either partition the block into one subblock or a million subblocks. Both altematives are undesirable.

A solution to this problem is artificially to impose an upper bound U and a lower bound L on the subblock size. There seem to be a limitless number of ways of doing this. Here are some examples:

Upper bound: Subdivide subblocks that are longer than U bytes at the points, U, 2U, 3U, and so on, where U is the chosen upperbound on subblock size.

Upper bound: Subdivide subblocks that are longer than U bytes at points determined by a secondary hash function.

Lower bound: Of the set of boundaries that bound subblocks less than L bytes long, remove those boundaries that are closer to their neighbouring boundaries than their neighbouring boundaries are to their neighbouring boundaries.

Lower bound: If the block is being scanned sequentially, do not place a boundary unless at least L bytes have been scanned since the previous boundary.

Lower bound: Of the set of boundaries that bound subblocks less than L bytes long, remove those boundaries that satisfy some secondary hash function.

Lower bound: Of the set of boundaries that bound subblocks less than L bytes long, remove randomly chosen boundaries until all the resulting subblocks are at least L bytes long.

Many other such schemes could be devised.

The Use of Multiple Partitionings

In most applications the use of just one partitioning into subblocks will be sufficient. However, in some applications there may be a need for more than one subblock partitioning. For example, in applications where channel space is expensive, it may be appropriate to partition each block of data in W different ways, using W different constraint functions $F_1 \ldots F_W$ where each function provides a different average subblock size. For example, four different partitions could be performed using functions that provide subblocks of average length 256 bytes, 1K, 10K, and 100K. By providing a range of different sizes of subblocks to choose from, such as organization could simultaneously indicate large blocks extremely efficiently, while still retaining fine-grained subblocks so that minor changes to the data do not result in voluminous updates (FIG. 8).

The efficiency of such a scheme could be improved by performing the partitioning all in one operation using increasing constraints on a single F. For example, one could use the example hash function described earlier, but use different values of the constant ρ to determine the different levels of subdivision. By choosing appropriately related values of ρ, the set of boundaries that could be produced by the different F could be arranged to be subsets of each other, resulting in a tree structure of subblocks. For example, values of ρ of 32, 64, and 128, and 256 could be used. FIG. 9 shows how the subblocks of four levels of the tree could relate to each other:

A further method could define the hash of a larger block to be the hash of the hashes of its component blocks.

Multiple partitionings could also be useful simply to provide a wider pool of subblocks to compare. For example, it may be appropriate to partition each block of data in W different ways using W different functions $F_1 \ldots F_W$ where each function yields roughly the same subblock sizes, but at different positions within the block.

Another technique would be to create an additional set of boundaries based on the boundaries provided by a hash function. For example, a fractal algorithm could be used to partition a block based upon some other partitioning provided by a function F.

Comparing Subblocks

In most applications of this invention, there will be a need at some stage to identify identical subblocks. This can be done in a variety of ways:

Compare the subblocks themselves.

Compare the hashes of the subblocks.

Compare identifies of the subblocks.

Compare references to the subblocks.

In most cases, the problem reduces to that of taking a group of subblocks of data and finding all subsets of identical subblocks. This is a well-solved problem and discussion of various solutions can be found in the following books:

Knuth D. E., "The Art of Computer Programming: Volume 1: Fundamental Algorithms". Addison Wesley, 1973.

Knuth D. E., "The Art of Computer Programming: Volume 3: Sorting and Searching", Addison Wesley, 1973.

In most cases, the problem is best solved by creating a data structure that maintains the subblocks, or references to the subblocks, in sorted order, and then inserts each subblock one at a time into the data structure. Not only does this identify all currently identical subblocks, but it also establishes a structure that can be used to determine quickly whether incoming subblocks are identical to any of those already held. The following data structures are described in the books referenced above and provide just a sample of the structures that could be used:

Hash tables.

Sorted trees (binary, N-ary, AVL).

Sorted linked lists.

Sorted arrays.

Of the multitude of solutions to the problem of matching blocks of data, one solution is worthy of special attention: the hash table. Hash tables consist of a (usually) finite array of slots into which values may be inserted. To add a value to a hash table, the value is hashed (using a hash function that is usually selected from the class of narrow hash functions) into a slot number, and the value is inserted into that slot. Later, the value can be retrieved in the same manner. Provisions must be made for the case where two data values, to be stored in the same table, hash to the same slot number.

Hash tables are likely to be of particular value in the implementation of this invention because:

They provide very fast (essentially constant time) access.

Many applications will need to calculate a strong one-way hash of each subblock, and a portion of this value can be used to index the hash table.

Particularly effective would be a hash table indexed by a portion of a strong one-way hash of the subblocks it stores, with each table entry containing (a) the strong one-way hash of the subblock, and (b) a pointer to the subblock stored elsewhere in memory.

The Use of Compression, Encryption, and Integrity Techniques

Aspects of the invention could be enhanced by the use of data compression, data encryption, and data integrity techniques. The applications of these techniques include, but are not limited to, the following applications:

Any subblock that is transmitted or represented in its raw form could alternatively be transmitted or represented in a compressed or encrypted form.

Subblocks could be compressed and encrypted before further processing by aspects of this invention.

Blocks could be compressed and encrypted before further processing by aspects of this invention.

Communications or representations could be compressed or encrypted.

Any component could carry additional checking information such as checksums or digests of the data in the component.

Ad-hoc data compression techniques could be used to further compress references and identities or consecutive runs of references and identities.

Storage of Variable-Length Subblocks on Disk

The division of data into subblocks of varying length presents some storage organization problems (if the subblocks are to be stored independently of each other), as most hardware disk systems are organized to store an array of fixed-length blocks (e.g. one million 512-byte blocks) rather than variable-length ones. Here are some techniques that could be used to tackle this problem:

Each subblock could be stored in an integral number of disk blocks, with some part of the last disk block being wasted. For randomly sized subblocks, this scheme will waste on average half a disk block per subblock.

Create a small subset of different bucket sizes (e.g. powers of two) and create arrays on the disk that pack collections of these buckets efficiently into the disk blocks. For example, if disk blocks were 512 bytes long, one could fairly efficiently pack five 200-byte buckets into an array of two disk blocks. Each subblock would be stored in the smallest bucket size that would hold the subblock, with the unused part of the bucket being wasted.

Treat the disk blocks as a vast array of bytes, and use well-established heap management techniques to manage the array. A sample of such techniques appears in pages 435–451 of the book:

Knuth D. E., "The Art of Computer Programming: Volume 1: Fundamental Algorithms", Addison Wesley, 1973.

The Use of Concurrency

Two processes are said to be concurrent if their execution takes place in some sense at the same time:

In interleaving concurrency, some or all of the operations performed by the two processes are interleaved in time, but the two processes are never both executing at exactly the same instant.

In genuine concurrency, some or all of the operations performed by the two processes are genuinely executed at the same instant. Implementations of the present invention could incorporate either form of concurrency to various degrees. In most of the aspects of the invention, some subset of the steps of each aspect could be performed concurrently. In particular (without limitation):

A block could be split into parts and the parts partitioned concurrently.

The processing of subblocks defined during a sequential partitioning of a block need not be deferred until the entire block has been partitioned. In particular, the hashes of already-defined subblocks could be calculated and compared while further subblocks are being defined.

Communicating entities that decompose and compose blocks could execute concurrently.

Where more than one block must be partitioned for processing, such partitioning could be performed concurrently.

Many more forms of concurrency within aspects of this invention could be identified.

Example: Partitioning a Block

We now present a simple example of how a block might be partitioned in practice. Consider the following block of bytes:

$b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8\ b_9\ \ldots$

In this example, an example hash function H will be used to partition the block. Boundaries will be represented by pairs such as $B_6|b_7$. We will assume that H returns a boolean value based on its argument and that a boundary is to be placed at each $b_i|b_i+1$ for which $H(b_i-2, b_i-1, b_i)$ evaluates to true.

As the hash function accepts 3 byte arguments, we start at $b_3|b_4$ and evaluate $H(b_1, b_2, b_3)$. This turns out to be false (for the purposes of example), so we move to $b_4|b_5$ and evaluate $H(b_2, b_3, b_4)$. This turns out to be true, so a boundary is placed at $b_4|b_5$. Next, we move to $b_5|b_6$ and evaluate $H(b_3, b_4, b_5)$. This turns out to be false so we move on. $H(b_4, b_5, b_6)$ is true so we place a boundary at $b_6|b_7$. This process continues until the end of the block is reached.

$b_1\ b_2\ b_3\ b_4 \mid b_5\ b_6 \mid b_7\ b_8\ b_9\ \ldots$

Some variations on this approach are:

Imposition of a lower bound L on subblock size by skipping ahead L bytes after placing a boundary.

Imposition of an upper bound U on block size by artificially placing a boundary if U bytes have been processed since the last boundary was placed.

Improving the efficiency of the hash calculations by using some part of the calculation of the has of the bytes at one position to calculate the hash at the next position. For example, it may be more efficient to calculate H(x,y,z) if H(*,x,y) has already been calculated. For example, the Internet IP checksum is organized so that a single running checksum value can be maintained, with bytes entering the window being added to the checksum, and bytes exiting the window being subtracted from the checksum.

Applying this algorithm in reverse, starting from the end of the block and working backwards.

Finding the subblock that encloses a particular point (chosen from anywhere within the block) by exploring in both directions from the point, looking for the nearest boundary in each direction.

Finding all subblock boundaries in one step of evaluating F for all position concurrently.

Example: Forming a Table of Hashes

Once a block has been partitioned, the hash of each subblock can be calculated to form a table of hashes (FIG. 24).

This table of hashes can be used to determine if a new subblock is identical to any of the subblocks whose hashes are in the table. To do this, the new subblock's hash is calculated and a check made to see if the hash is in the table.

In FIG. 24, the table of hashes looks like an array of hashes. However, the table of hashes could be stored in a wide variety of data structures (e.g. hash tables, binary trees).

Example Application: A File Comparison Utility

As the invention provides a new way of finding similarities between large volumes of data, it follows that it should find some application in the comparison of data.

In one aspect, the invention could be used to determine the broad similarities between two files being compared by a file comparison utility. The utility would partition each of the two files into subblocks, organize the hashes of the subblocks somehow (e.g. using a hash table) to identify all identical subblocks, and then use this information as a framework for reporting similarities and differences between the two files.

In a similar aspect, the invention could be used to find similarities between the contents of large numbers of files in a file system. A utility incorporating the invention could read each file in an entire file system, partition each into subblocks and then insert the subblocks (or hashes of the subblocks) into one huge table (e.g. implemented by a hash table or a binary tree). If each entry in the table carried the name of the file containing it as well as the position of the subblock within the file, the table could later be used to identify those files containing identical portions of data.

If, in addition, a facility was added for recording and comparing the hashes of the entire contents of files and directory trees, a utility could be constructed that could identify all largely similar structures within a file system. Such a utility would be immensely useful when (say) attempting to merge the data on several similar backup tapes.

Example Application: A Fine-Grained Incremental Backup System

In a fine-grained incremental backup system, two entities E1 and E2 (e.g. two computers on a network) wish to repeatedly backup a file X at E1 such that the old version of the file Y at E2 will be updated to become a copy of the new version of the file X at E1 (without modifying X). The system could work as follows.

Each time E1 performs a backup operation, it partitions X into subblocks and writes the hashes of the subblocks to a shadow file S. It might also write a hash of the entire contents of X to the shadow file. After the backup has been completed, X will be the same as Y and so the shadow file S will correspond to both X and Y. Once X is again modified (during the normal operation of the computer system), S will correspond only to Y. S can then be used during the next backup operation.

To perform the backup, E1 compares the hash of Y (stored in S) against the hash of X to see if X has changed (it could also use the modification date file attribute of the file). If X hasn't changed, there is no need to perform any further backup action. If X has changed, E1 partitions X into subblocks, and compares the hashes of these subblocks with the hashes in the shadow file S, so as to find all identical hashes. Identical hashes identify identical subblocks in Y that can be transmitted by reference. E1 then transmits the file as a mixture of raw subblocks and references to subblocks whose hashes appear in S and which are therefore known to appear as subblocks in Y. E1 can also transmit references to subblocks already transmitted. References can take many forms including (without limitation):

A hash of the subblock.

The number of the subblock in the list of subblocks in Y.

The number of a subblock previously transmitted.

A range of any of the above.

Throughout this process, E1 can be constructing the new shadow file corresponding to X. FIG. 25 illustrates the backup process.

To reconstruct X from Y and D (the incremental backup information being sent from E1), E2 partitions Y into subblocks said calculates the hashes of the subblocks (It could do this in advance during the previous backup). It then processes the incremental backup information, copying subblocks that were transmitted raw and looking up the references either in Y or in the part of X already reconstructed.

Because information need only flow from E1 to E2 during the backup operation, there is no need for E1 and E2 to perform the backup operation concurrently. E1 can perform its side of the backup operation in isolation, producing an incremental backup file that can be later processed by E2.

There is a tradeoff between 1) the approximate ratio between the size of each file and that of its shadow, and 2) the mean subblock size. The higher the mean subblock size (as determined by the partitioning method used), the fewer subblocks per unit file length, and hence the shorter the shadow size per unit file length. However, increasing means subblock sizes implies increasing the granularity of backups which can cause an increase in the size of the incremental backup file. There is also a tradeoff between the shadow file size and the hash width. A shadow file that uses 128-bit hashes will be about twice as long as one that uses 64-bit hashes. All these tradeoffs must be considered closely when choosing an implementation.

Bytes	Description
16	MD5 digest of the file Y corresponding to this shadow file.
16	MD5 digest of the first subblock in Y.
16	MD5 digest of the second subblock in Y.
..	. . .
16	MD5 digest of the last subblock in Y.
16	MD5 digest of the rest of this shadow file.

The first field contains the MD5 digest (a form of cryptographic hash) of the entire contents of Y. This is included so that it can be copied to the incremental backup file so as to provide a check later that the incremental backup file is not being applied to the wrong version of Y. It could also be used to determine if any change has been made to X since the previous backup Y was taken. The first field is followed by a list of the MD5 digests of the subblocks in Y in the order in which they appear in Y. Finally, a digest of the contents of the shadow file (less this field) is included at the end so as to enable the detection of any corruption of the shadow file.

The format of the incremental backup file is as follows:

Bytes	Description
16	MD5 digest of Y.
16	MD5 digest of X.
..	Zero or more ITEMS.
16	MD5 digest of the rest of the incremental backup file.

The first two fields of the incremental backup file contain the MD5 digest of the old and new versions of the file. The hash of the new version X is calculated directly from X. The hash of the old version is obtained from the first field of the shadow file. These two values enable the remote backup entity E2 to check that:

The backup file Y (to be updated) is identical to the one from which the shadow file was generated.

The reconstructed X is identical to the original X.

The two checking fields are followed by a list of items followed by a checking digest of the rest of the incremental backup file.

Each item in the list of items describes one or more subblocks in the list of subblocks that can be considered to

constitute X. There are three kinds of item, and each item commences with a byte having a value one, two, or three to indicate the kind of item. Here is a description of the content of each of the three kinds of item:

1. The 32-bit index of s subblock in Y. Because E2 possesses Y, it can partition Y itself to construct the same partitioning that was used to create the shadow file. Thus E1 doesn't need to send the hash of any subblock that is in both X and Y. Instead, it need only send the index of the subblock in the list of subblocks constituting Y. This list is represented by the list of hashes in the shadow file S. As 32-bits is wide enough for an index in practice, the saving gained by communicating a 32-bit index instead of a hash is 98 bits for each such item.
2. A pair of 32-bit numbers being the index of the first and last subblock of a range of subblocks in Y. Old and new versions of files often share large contiguous ranges of subblocks. The use of this kind of item allows such ranges to be represented using just 64 bits instead of a long run of instances of the first kind of item.
3. A 32-bit value containing the number of bytes in the subblock, followed by the raw content of the subblock. This kind of item is used if the subblock to be transmitted is not present in Y.

In the implementation, all the values are coded in little-endian form. Big-endian could be used equally as well.

The existing implementation could be further optimized by (without limitation):

Adding an additional kind of item that refers to subblocks in X already transmitted;

Adding an additional kind of item that refers to ranges of subblocks in X already transmitted;

Employing data compression techniques to compress the raw blocks in the third kind of item.

Using the first hash in the shadow file to check to see if the entire file has changed at all before performing the backup process described above.

Replacing hashes in S of subblocks in Y by references to other hashes in S (where the hashes (and hence subblocks) are identical). Repeated runs of hashes could also be replaced by pointers to ranges of hashes.

The scheme described above has been described in terms of a single file. However, the technique could be applied repeatedly to each of the files in a file system, thus providing a way to back up an entire file system. The shadow information for each file in the file system could be stored inside a separate shadow file for each file, or in a master shadow file containing the shadows for one or more (or all) files in the file system.

Although most redundancy in a file system is likely to be found within different versions of each file, there may be great similarities between versions of different files. For example, if a file is renamed, the "new" file will be identical to the "old" file. Such redundancy can e catered for by comparing the hashes of all the files in the old and new versions of a file system. In addition, similarities between different parts of different files can be exploited by comparing the hashes of subblocks of each file to be backed up with the hashes of the subblocks of the entire old version of the file system.

If E2 has lots of space, a further improvement could be for E1 to retain the shadows of all the previous versions of the file system, and for E2 to retain copies of all the previous versions of the file system. E1 could then refer to every block it has ever seen. This technique could also be applied on a file-by-file basis.

In a further variant, the dependence on the ordering of subblocks could be abandoned, and E1 could simply keep a shadow file containing a list of the hashes of all the subblocks in the previous version (or versions) of the file or file system. E2 would then need to record only a single copy of each unique subblock it has ever received from E1.

Aspects of the backup application described in this section can be integrated cleanly into existing backup architecture by deploying the new mechanisms within the framework of existing ones. For example, the traditional methods for determining if a file has changed since the last backup (modification date, backup date and so on) can be used to see if a file needs to be backed up at all, before applying the new mechanisms.

Example Application: A Low-Redundancy File System

We now present an example of a low-redundancy file system that attempts to avoid storing different instances of the same data more than once. In this example, the file system is organized as shown in FIG. **26**.

The bottom layer consists of a collection of unique subblocks of varying length that are stored somewhere on the disk. The middle layer consists of a hash table containing one entry for each subblock. Each entry consists of a cryptographic hash of the subblock, a reference count for the subblock, and a pointer to the subblock on disk. The hash table is indexed by some part of the cryptographic hash (e.g. the bottom 16 bits). Although a hash table is used in this example, many other data structures (e.g. a binary tree) could also be used to map cryptographic hashes to subblock entries. It would also be possible to index the subblocks directly without the use of cryptographic hashes.

The top layer consists of a table of files that binds filenames to lists of subblocks, each list being a list of indexes into the hash table. The reference count of the hash table records the number of references to the subblock that appear in the entire set of files in the file table. The issue of hash table "overflow" can be addressed using a variety of well-known overflow techniques such as that of attaching a linked list to each hash slot.

When a file is read, the list of hash table indexes is converted to pointers to subblocks of data using the hash table. If random access to the file is required, extra information about the length of the subblocks could be added to the file table and/or hash table so as to speed access.

Writing a file is more complicated. During a sequential write, the data being written is buffered until a subblock-boundary is reached (as determined by whatever boundary function is being used). The cryptographic hash of the new subblock is then calculated and used to look up the hash table. If the subblock is unique (i.e. there is no entry for the cryptographic hash), it is added to the data blocks on the disk and an entry is added to the hash table. A new subblock number is added to the list of blocks in the file table. If, on the other hand, the subblock already exists, the subblock need not be written to disk. Instead, the reference count of the already-existing subblock is incremented, and the subblock's hash table index is added to the list of blocks in the file's entry in the file table.

Random access writes are more involved, but essentially the same principles apply.

If a record were kept of subblocks created since the last backup, backing np this file system could be very efficient indeed.

One enhancement that could be made is to exploit unused disk space. Instead of automatically ignoring or overwriting

subblocks whose reference count has dropped to zero, the low-redundancy file system could move them to a pool of unused subblocks. These subblocks, while not present in any file, could still form part of the subblock pool referred to when checking to see if incoming subblocks are already present in the file system. The space consumed by subblocks in the unused subblock pool would be recycled only when the disk was full. In the steady state, the "unused" portion of the disk would be filled by subblocks in the unused subblock pool.

Although this section has specifically described a low-redundancy file system, this aspect of the invention is really a general purpose storage system that could be applied at many levels and in many roles in information processing systems. For example:

The technique could be used to implement a low-redundancy virtual memory system. The contents of memory could be organized as a collection of subblocks.

The technique could be used to increase the efficiency of an on-chip cache.

Example Application: A Communication System

A method is now presented for reducing redundant transmissions in communications systems. Consider two entities E1 and E2, where E1 must transfer a block of data X to E2. E1 and E2 need never have communicated previously with each other.

The conventional way to perform the transmission is simply for E1 to transmit X to E2. However, here E1 first partitions X into subblocks and calculates the hash of each subblock using a hash function. It then transmits the hashes to E2. E2 then looks up the hashes in a table of hashes of all the subblocks it already possesses. E2 then transmits to E1 information (e.g. a list of subblock numbers) identifying the subblocks in X that E2 does not already possess. E1 then transmits just those subblocks.

Another way to perform the transaction would be for E2 to first transmit to E1 the hashes of all the subblocks it possesses (or perhaps a well chosen subset of them). E1 could then transmit references to subblocks in X already known to E2 and the actual contents of subblocks in X not known to E2. This scheme could be more efficient than the earlier scheme in cases where E2 possesses less subblocks than there are in X.

Another way to perform the transaction would be for E1 and E2 to conduct a more complicated conversation to establish which subblocks E2 possesses. For example, E2 could send E1 the hashes of just some of the subblocks it possesses (perhaps the most popular ones). E1 could then send to E2 the hashes of other subblocks in X. E2 could then reply indicating which of those subblocks it truly does not possess. E1 could then send to E2 the subblocks in X not possessed by E2.

In a more sophisticated system, E1 and E2 could keep track of the hashes of the subblocks possessed by the other. If either entity ever sent (for whatever reason) a reference to a subblock not possessed by the other entity, the latter entity could simply send back a request for the subblock to be transmitted explicitly and the former entity could send the requested subblock.

The communication application described above considers the case of just two communicants. However, there is no reason why the scheme could not be generalized to cover more than two communicants communicating with each other in private and in public (using broadcasts). For example, to broadcast a block, a computer C_1 could broadcast a list of the hashes of the block's subblocks. Computers $C_2 \ldots C_N$ could then each reply indicating which subblocks they do not already possess. C_1 could then broadcast subblocks that many of the other computers do not possess, and send the subblocks missing from only a few computers to those computers privately.

All these techniques have the potential to greatly reduce the amount of information transmitted between computers.

These techniques would be very efficient if they were implemented on top of the file system described earlier, as the file system would already have performed the work of organizing all the data it possesses into indexed subblocks. The potential savings in communication that could be made if many different computer systems shared the same subblock partitioning algorithm suggests that some form of universal standardization on a particular partitioning method would be a worthy goal.

Example Application: A Subblock Server

Aspects of the invention could be used to establish a subblock server on a network so as to reduce network traffic. A subblock server could be located in a busy part of a network. It would consist of a computer that breaks each block of data it sees into subblocks, hashes the subblocks, and then stores them for future reference. Other computers on the network could send requests to the server for subblocks, the requests consisting of the hashes of subblocks the server might possess. The server would respond to each hash, returning either the subblock corresponding to the hash, or a message stating that the server does not possess a subblock corresponding to the hash.

Such a subblock server could be useful for localizing network traffic on the Internet. For example, if a subnetwork (even a large one for (say) an entire country) placed a subblock server on each of its major Internet connections, then (with the appropriate modification of various protocols) much of the traffic into the network could be eliminated. For example, if a user requested a file from a remote host on another network, the user's computer might issue the request and receive, in reply, not the file, but the hashes of the file's subblocks. The user's computer could then send the hashes to the local subblock server to see if the subblocks are present there. It would receive the subblocks that are present and then forward a request for the remaining subblocks to the remote host. The subblock server might notice the new subblocks flowing through it and archive them for future reference. The entire effect could be to eliminate most repeated data transfers between the subnetwork and the rest of the Internet. However, the security implications of schemes such as these would need to be closely investigated before there were deployed.

A further step could be to create "virtual" subblock servers that store the hashes of subblocks and their location on the Internet rather than the subblocks and their hashes.

I claim:

1. A method for organizing a block b of digital data for storage, communication, or comparison, comprising the step of:

partitioning said block b into a plurality of subblocks at at least one position k|k+1 within said block,

for which b[k−A+1 . . . k+B] satisfies a predetermined constraint, and

wherein A and B are natural numbers.

2. The method of claim 1, wherein the constraint comprises the hash of at least a portion of b[k−A+1 . . . k+B].

3. The method of claim 1, further comprising the step of:

locating the nearest subblock boundary on a side of a position p|p+1 within said block, said locating step comprising the step of:

evaluating whether said predetermined constraint is satisfied at each position k|k+1 for increasing or decreasing k,

wherein k starts with the value p.

4. The method of claim 1, wherein at least one bound is imposed on the size of at least one of said plurality of subblocks.

5. The method of claim 1, wherein additional subblocks are formed from at least one group of subblocks.

6. The method of claim 1, wherein an additional hierarchy of subblocks is formed from at least one group of contiguous subblocks.

7. The method of claim 1, further comprising the step of:

calculating the hash of each of at least one of said plurality of subblocks.

8. The method of claim 1, further comprising the step of:

forming a projection of said block, being an ordered or unordered collection of elements, wherein each element consists of a subblock, an identity of a subblock, or a reference of a subblock.

9. The method of claim 1, wherein said subblocks are compared by comparing the hashes of said subblocks.

10. The method of claim 1, wherein subsets of identical subblocks within a group of one or more subblocks are found by inserting each subblock, an identity of each subblock, a reference of each subblock, or a hash of each subblock into a data structure.

11. A method for comparing one or more blocks, comprising the steps of:

organizing a block b of digital data for the purpose of comparison, comprising the step of:

partitioning said block b into a plurality of subblocks at at least one position k|k+1 within said block;

for which b[k−A+1 . . . k+B] satisfies a predetermined constraint; and

wherein A and B are natural numbers,

forming a projection of each said block, being a collection of elements, wherein each element comprises a selected one of a subblock, an identity of a subblock, and a reference of a subblock, and

comparing the elements of said projections of said blocks.

12. A method for representing one or more blocks comprising a collection of subblocks and block representatives which are mapped to lists of entries which identify subblocks; said method comprising the step of modifying one of said blocks including the steps of:

partitioning said block into a plurality of subblocks at at least one position k|k+1 within said block, for which b[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

adding to said collection of subblocks zero or more subblocks which are not already in said collection, and

updating said subblock list associated with said modified block.

13. A method for representing one or more blocks comprising a collection of subblocks and block representatives which are mapped to lists of entries which identify subblocks; said method comprising the step of modifying one of said blocks including the steps of:

partitioning said block into a plurality of subblocks at at least one position k|k+1 within said block, for which b[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

removing from said collection of subblocks zero or more subblocks, and

updating said subblock list associated with said modified block.

14. A method for representing one or more blocks comprising a collection of subblocks and block representatives which are mapped to lists of entries which identify subblocks; said method comprising the step of modifying one of said blocks including the steps of:

partitioning said block into a plurality of subblocks at at least one position k|k+1 within said block, for which b[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

adding to said collection of subblocks zero or more subblocks that are not already in the collection,

removing from said collection of subblocks zero or more subblocks, and

updating said subblock list associated with said modified block.

15. A method for an entity E1 to communicate a block X to E2 where E1 possesses the knowledge that E2 possesses a group of Y subblocks $Y_1 \ldots Y_m$, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

transmitting from E1 to E2 the contents of zero or more subblocks in X_1 and the remaining subblocks as references to subblocks in $Y_1 \ldots Y_m$, and to subblocks transmitted.

16. A method for an entity E1 to communicate one or more subblocks of a group X of subblocks $X_1 \ldots X_n$ to E2 where E1 possesses the knowledge that E2 possesses a block Y, comprising the steps of:

partitioning said block Y into a plurality of subblocks $Y_1 \ldots Y_m$ at at least one position k|k+1 within said block, for which Y[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

transmitting from E1 to E2 the contents of zero or more subblocks in X, and the remaining subblocks as references to subblocks in Y, and to subblocks already transmitted.

17. A method for an entity E1 to communicate a block X to E2 where E1 possesses the knowledge that E2 possesses a block Y, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

partitioning said block Y into a plurality of subblocks $Y_1 \ldots Y_m$ at at least one position k|k+1 within said block, for which Y[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

transmitting from E1 to E2 the contents of zero or more subblocks in X, and the remaining subblocks as references to subblocks in Y, and to subblocks already transmitted.

18. A method for constructing a block D from a block X and a group Y of subblocks $Y_1 \ldots Y_m$ such that X can be constructed from Y and D, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

constructing D from a selected at least one of:
the contents of zero or more subblocks in X,
references to zero or more subblocks in Y, and
references to zero or more subblocks in D.

19. A method for constructing a block D from a group X of subblocks $X_1 \ldots X_n$ and a block Y such that X can be constructed from Y and D, comprising the steps of:

partitioning said block Y into a plurality of subblocks $Y_1 \ldots Y_m$ at at least one position k|k+1 within said block, for which Y[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

constructing D from a selected at least one of:
the contents of zero or more subblocks in X,
references to zero or more subblocks in Y, and
references to zero of more subblocks in D.

20. A method for constructing a block D from a block X and a block Y such that X can be constructed from Y and D, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

partitioning said block Y into a plurality of subblocks $Y_1 \ldots Y_m$ at at least one position k|k+1 within said block, for which Y[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

constructing D from a selected at least one of:
the contents of zero or more in X,
references to zero or more subblocks in Y, and
references to zero or more subblocks in D.

21. A method for constructing a block D from a block X and a projection Y said projection comprising a collection of elements wherein said elements comprises a subblock in Y, an identity of a subblock in Y, or a reference of a subblock in Y, such that X can be constructed from Y and D, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

constructing D from a selected at least one of:
the contents of zero or more in X,
references to zero or more subblocks in Y, and
references to zero or more subblocks in D.

22. A method for constructing a block X from a block Y and a block D, comprising the steps of:

partitioning said block Y into a plurality of subblocks $Y_1 \ldots Y_m$ at at least one position k|k+1 within said block, for which Y[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

constructing X from D and Y by constructing the subblocks of X based on a selected at least one of:
subblocks contained within D,
references in D to subblocks in Y, and
references to D to subblocks in D.

23. A method for constructing a group X of subblocks $X_1 \ldots X_n$ from a block Y and a block D, comprising the steps of:

partitioning said block Y into a plurality of subblocks $Y_1 \ldots Y_m$ at at least one position k|k+1 within said block, for which Y[k−A+1 . . . k+b] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

constructing $X_1 \ldots X_n$ from D and Y based on a selected at least one of:
subblocks contained within D,
references in D to subblocks in Y, and
references to D to subblocks in D.

24. A method for communicating a data block X from one entity E1 to another entity E2, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

transmitting from E1 to E2 an identity of at least one subblock,

transmitting from E2 to E1 information communicating the presence or absence at E2, and

transmitting from E1 to E2 at least the subblocks identified as not being present at E2.

25. A method for communicating a block X from one entity E1 to another entity E2, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

transmitting from E2 to E1 information communicating the presence or absence at E2 of members of a group Y of subblocks $Y_1 \ldots Y_m$, and

transmitting from E1 to E2 the contents of zero or more subblocks in X, and the remaining subblocks as references to subblocks in $Y_1 \ldots Y_m$ and to subblocks already transmitted.

26. A method for an entity E2 to communicate to an entity E1 the fact that E2 possesses a block Y, comprising the steps of:

partitioning said block Y into a plurality of subblocks $Y_1 \ldots Y_m$ at at least one position k|k+1 within said block, for which Y[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers, and

transmitting from E2 to E1 references of the subblocks $Y_1 \ldots Y_m$.

27. A method for an entity E1 to communicate a subblock X_1 to an entity E2, comprising the steps of:

partitioning said block X into a plurality of subblocks $X_1 \ldots X_n$ at at least one position k|k+1 within said block, for which X[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers,

transmitting from E2 to E1 an identity of X_i,

transmitting X_i from E1 to E2.

28. An apparatus for organizing a block b of digital data for storage, communication, or comparison, comprising

means for partitioning said block b into a plurality of subblocks at at least one position k|k+1 within said block, for which b[k−A+1 . . . k+B] satisfies a predetermined constraint, and wherein A and B are natural numbers.

29. The apparatus of claim 28, in which the constraint comprises the hash of some or all of b[k−A+1 . . . k+B].

30. The apparatus of claim 28, further comprising

means for locating the nearest subblock boundary on a side of a position p|p+1 within said block, said means for locating comprising:

means for evaluating whether said predetermined constraint is satisfied at each position k|k+1 for increasing or decreasing k,

wherein k starts with the value p.

* * * * *

EXHIBIT B

QUINN EMANUEL URQUHART OLIVER & HEDGES, LLP
Claude M. Stern (Bar No. 96737)
Todd M. Briggs (Bar No. 209282)
555 Twin Dolphin Drive, Suite 560
Redwood Shores, California 94065-2139
Telephone: (650) 801-5000
Facsimile: (650) 801-5100
Email: claudestern@quinnemanuel.com
toddbriggs@quinnemanuel.com

Attorneys for Defendant and Counterclaimant,
RIVERBED TECHNOLOGY, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

QUANTUM CORPORATION,

Plaintiff,

vs.

RIVERBED TECHNOLOGY, INC.,

Defendant.

RIVERBED TECHNOLOGY, INC.,

Counterclaimant,

vs.

QUANTUM CORPORATION,

Counterdefendant.

CASE NO. C 07-4161 WHA

RIVERBED'S FIRST AMENDED ANSWER AND COUNTERCLAIMS FOR:

INFRINGEMENT OF U.S. PATENT NO. 7,116,249; AND

DECLARATORY JUDGMENT OF NON-INFRINGEMENT AND INVALIDITY OF U.S. PATENT NO. 5,990,810

DEMAND FOR JURY TRIAL

Counterclaim Plaintiff Riverbed Technology, Inc. ("Riverbed") by and through its undersigned counsel, respectfully submits Riverbed's First Amended Answer and Counterclaims For: Infringement of U.S. Patent No. 7,116,249; and Declaratory Judgment of Non-Infringement and Invalidity of U.S. Patent No. 5,990,810 pursuant to Federal Rule of Civil Procedure 15(a). On October 29, 2007, Riverbed filed its Answer, Affirmative Defenses and Counterclaims in response to the Complaint for Patent Infringement ("Complaint") of Plaintiff and Counterclaim Defendant Quantum Corporation, which was filed on August 14, 2007 and served on October 9, 2007. Quantum has not responded to Riverbed's Answer, Affirmative Defenses and Counterclaims. Riverbed's First Amended Answer and Counterclaim amends Riverbed's Answer and Counterclaim by including an additional counterclaim for infringement of Riverbed's U.S. Patent No. 7,116,249 ("the '249 Patent") against Quantum.

RIVERBED'S ANSWER TO QUANTUM'S COMPLAINT

Riverbed answers Quantum's Complaint as follows:

PARTIES

1. Riverbed lacks sufficient knowledge or information to form a belief as to the truth of the allegations in paragraph 1, and therefore denies the allegations in this paragraph.

2. Riverbed admits the allegations in paragraph 2.

JURISDICTION AND VENUE

3. Riverbed admits the allegations in paragraph 3.

4. Riverbed admits that this Court has personal jurisdiction over Riverbed.

5. Riverbed admits the allegations in paragraph 5.

BACKGROUND

6. Riverbed admits that Ross Neil Williams is named as an inventor of U.S. Patent No. 5,990,810 ("the '810 patent"), but denies that the '810 patent is directed to a new, useful and non-obvious method and apparatus for partitioning a block of data into subblocks and for storing and communicating such subblocks and the remaining allegations in paragraph 6.

7. Riverbed admits that the '810 patent is entitled "Method For Partitioning A Block Of Data Into Subblocks And For Storing And Communicating Such Subblocks" and that the issue date on the face of the patent is November 23, 1999, but denies that that '810 was duly and lawfully issued by the United States Patent and Trademark Office. Riverbed admits that a purported copy of the '810 patent was attached as Exhibit A to the Complaint.

8. Riverbed lacks sufficient knowledge or information to form a belief as to the truth of the allegations in paragraph 8, and therefore denies the allegations in this paragraph.

9. Riverbed denies the allegations in paragraph 9.

COUNT FOR INFRINGEMENT OF U.S. PATENT NO. 5,990,810

10. Riverbed incorporates its answers to paragraphs 1 through 9 of the Complaint as if set forth fully herein.

11. Riverbed denies the allegations in paragraph 11

12. Riverbed denies the allegations in paragraph 12.

13. Riverbed denies the allegations in paragraph 13.

14. Riverbed denies the allegations in paragraph 14.

15. Riverbed denies the allegations in paragraph 15.

16. Riverbed denies the allegations in paragraph 16.

PRAYER FOR RELIEF

Riverbed denies that Quantum is entitled to any of the relief sought by its Prayer for Relief.

AFFIRMATIVE DEFENSES

FIRST AFFIRMATIVE DEFENSE: Non-Infringement

Riverbed does not make, use, sell, offer for sale, or import into the United States, and has not made, used, sold, offered for sale or imported into the United States any products or methods that infringe any valid claim of the '810 patent either directly, indirectly, contributorily, through the doctrine of equivalents, or otherwise, and has not induced others to infringe the '810 patent.

SECOND AFFIRMATIVE DEFENSE: Invalidity

The '810 patent is invalid for failure to meet the conditions for patentability set forth in Title 35 of the United States Code, including, but not limited to, 35 U.S.C. §§ 101, 102, 103, and 112.

THIRD AFFIRMATIVE DEFENSE: Laches and Equitable Estoppel

Quantum's claims are barred by the doctrines of laches and equitable estoppel.

FOURTH AFFIRMATIVE DEFENSE: Failure to Mark

On information and belief, prior to the filing of the Complaint against Riverbed, the owner of the '810 patent failed to properly mark its products or services and/or licensees of the '810 patent failed to properly mark their products or services and the owner of the '810 patent did not provide Riverbed with notification of any alleged infringement of the '810 patent. Under 35 U.S.C. § 287(a), Quantum is therefore barred from recovering damages for any alleged infringement of the '810 patent by Riverbed prior to the filing of the Complaint.

FIFTH AFFIRMATIVE DEFENSE: Standing

On information and belief, Quantum lacks standing to assert infringement of the '810 patent because it was not the legal or equitable owner of the '810 patent and/or it did not have all substantial rights in the '810 patent at the time the suit was filed.

SIXTH AFFIRMATIVE DEFENSE: Adequate Remedy at Law

Quantum is not entitled to injunctive relief because any alleged injury to Quantum is not immediate or irreparable, and Quantum has an adequate remedy at law.

SEVENTH AFFIRMATIVE DEFENSE: Failure to State a Claim

The Complaint fails, in whole or in part, to state a claim on which relief may be granted.

RIVERBED'S COUNTERCLAIMS

Riverbed hereby complaints against Quantum as follows:

PARTIES

1. Riverbed is a corporation organized and existing under the laws of the state of Delaware, having a principal place of business at 199 Fremont Street, San Francisco, California, 94105. Riverbed is qualified and duly authorized to do business in the State of California.

2. Riverbed is the technology and market leader in wide area data services (WDS). Riverbed designs, builds, and sells devices that facilitate the efficient sharing of computer data over extended, or wide-area, computer networks. Riverbed's technology is used to, among other things, improve the performance of everyday computer applications, such as email, file sharing, and document management, by a broad range of companies. Riverbed has developed and patented pioneering technologies applicable to wide area data services applications as well as other applications, such as data storage applications.

3. Quantum is a Delaware corporation organized and existing under the laws of the state of Delaware, having a principle place of business at 1650 Technology Drive, Suite 700, San Jose, CA 95110.

4. Quantum designs, builds, and sells data storage products that utilize Quantum's "data de-duplication" technology. Such products include but are not limited to Quantum's DXi-Series disk backup and replication systems. Quantum's DXi-Series products (and likely other products as well) segment data prior to storage and in doing so infringe Riverbed's '249 patent, which covers systems and methods for storing data.

JURISDICTION AND VENUE

5. This Court has subject matter jurisdiction over these Counterclaims pursuant to 28 U.S.C. §§ 1331 and 1338 because this action includes patent infringement claims under the patent laws of the United States, 35 U.S.C. § 1 *et seq.*, and pursuant to 28 U.S.C. §§ 2201 and 2202 because this action includes declaratory judgment claims.

6. By filing its Complaint, Quantum consented to the personal jurisdiction of this Court. Quantum is also subject to personal jurisdiction in this Court because, *inter alia*, and upon

information and belief, Quantum directly and through agents regularly does, solicits and transacts business in the Northern District of California and elsewhere in the state of California, including business with respect to the goods and services that are the subject of this action.

7. Venue is proper in this judicial district pursuant to 28 U.S.C. §§ 1391 and 28 U.S.C. § 1400.

EXISTENCE OF ACTUAL CONTROVERSY

8. Quantum alleges in its Complaint that it is an exclusive licensee of U.S. Patent No. 5,990,810 ("the '810 patent").

9. Quantum alleges in its Complaint that Riverbed has manufactured, sold, offered to sell and/or imported Riverbed Steelhead products that infringe the '810 patent.

10. Quantum alleges in its Complaint that the '810 patent is valid and meets all the requirements of 35 U.S.C. § 1 *et seq.*

FIRST COUNTERCLAIM

(Infringement of United States Patent No. 7,116,249)

11. Riverbed incorporates the allegations of paragraphs 1-10 as if fully set forth in this paragraph.

12. United States Patent No. 7,116,249 ("the '249 patent") is titled "Content-Based Segmentation Scheme for Data Compression in Storage and Transmission Including Hierarchical Segment Representation" and was duly and legally issued on October 3, 2006. Riverbed is the legal owner of the '249 patent and has the right to sue for infringements of this patent. A true and correct copy of the '249 patent is attached hereto as Exhibit A.

13. Quantum has directly infringed and continues to directly infringe the '249 patent by making, using, offering for sale, selling and/or causing to be made, used, offered for sale or sold products, systems, and/or services that practice the inventions of the '249 patent in violation of 35 U.S.C. § 271(a). Specifically, Quantum offers a series of infringing data storage products that utilize its "data de-duplication" technology including but not limited to its DXi7500, DXi5500, and DXi3500 series. Riverbed reserves the right to identify additional Quantum products that infringe claims of the '249 patent.

14. Quantum has induced others to infringe and continues to induce others to infringe the '249 patent in violation of 35 U.S.C. § 271(b), and has contributorily infringed and continues to contributorily infringe the '249 patent in violation of 35 U.S.C. § 271(c).

15. Quantum infringes numerous claims in the '249 patent including but not limited to claims 1 and 19, which recite:

> 1. A method for storing data, comprising:
>
> receiving a file from a client at a server;
>
> segmenting the file into one or more segments;
>
> forming a list, comprising:
>
> determining whether each segment is present in a segment store, wherein a segment that is present in the segment store has an assigned reference label;
>
> for each segment present in the segment store, adding to the list the assigned reference label; and
>
> for each segment not present in the segment store, assigning a reference label to the segment, storing the segment and the reference label, and adding the reference label to the list; and
>
> storing an association between the file and the list.
>
> 19. A system for storing files, comprising:
>
> a front-end file system for receiving from a client a file command for a file, wherein the front-end file system includes a front-end file server;
>
> a back-end storage system including logic to segment the file; and a segment store for storing the segments;
>
> a network file system interface for sending or receiving contents of the file between the front-end file system and the back-end storage system.

Riverbed reserves the right to identify additional Quantum products as infringing and to identify additional claims of the '249 patent as being infringed.

16. Quantum's infringement of the '249 patent has caused damage to Riverbed, and Riverbed is entitled to recover from Quantum the damages sustained by Riverbed as a result of its wrongful acts in an amount subject to proof at trial.

17. Quantum's infringement of the '249 patent will continue to damage Riverbed's business, causing irreparable harm for which there is no adequate remedy at law, unless it is enjoined by this Court.

18. Quantum's infringement of the '249 patent has been willful and deliberate, in that, on information and belief, Quantum has known of the '249 patent and has continued to infringe the claims of the '249 patent, entitling Riverbed to enhanced (and up to treble) damages under 35 U.S.C. § 284 and to attorneys' fees and costs incurred in prosecuting this action under 35 U.S.C. § 285.

SECOND COUNTERCLAIM

(Declaratory Judgment of Non-Infringement)

19. Riverbed incorporates the allegations in paragraphs 1 through 10 of these counterclaims as if fully set forth herein.

20. Riverbed's products do not infringe, either directly or indirectly, any claim of the '810 patent.

21. Quantum's wrongful allegations of infringement of the '810 patent and Riverbed's denial of those allegations create the existence of an actual controversy under 28 U.S.C. §§ 2201 and 2202. Riverbed seeks a judgment that it does not infringe any claim of the '810 patent.

SECOND COUNTERCLAIM

(Declaratory Judgment of Invalidity)

22. Riverbed incorporates the allegations in paragraphs 1 through 10 of these counterclaims as if fully set forth herein.

23. The '810 patent is invalid for failure to satisfy one or more of the conditions of patentability set forth Title 35 of the United States Code, including, but not limited to, 35 U.S.C. §§ 101, 102, 103 and 112.

24. Quantum's wrongful allegations of validity of the '810 patent and Riverbed's denial of those allegations create the existence of an actual controversy under 28 U.S.C. §§ 2201 and 2202. Riverbed seeks a judgment that every claim of the '810 patent is invalid.

EXCEPTIONAL CASE

This is an exceptional case entitling Riverbed to an award of its attorneys' fees incurred in connection with defending and prosecuting this action pursuant to 35 U.S.C. § 285, as a result of, *inter alia*, Quantum's assertion of the '810 patent against Riverbed with the knowledge that the '810 patent is not infringed and/or invalid.

PRAYER FOR RELIEF

WHEREFORE, Riverbed prays for:

A. Judgment that the '249 Patent is valid and enforceable;

B. Judgment that Quantum infringes the '249 Patent under 35 U.S.C. § 271 (a), (b), and (c);

C. Judgment that Quantum's infringement of the '249 Patent under 35 U.S.C. § 271 (a), (b), and (c) has been willful and deliberate, and for an award to Riverbed of treble damages pursuant to 35 U.S.C. § 284;

D. An award to Riverbed of damages adequate to compensate Riverbed for Quantum's infringement of the '249 Patent, but in no event less than a reasonable royalty together with interests and costs;

E. An entry of preliminary and permanent injunctive relief enjoining and restraining Quantum and its officers, directors, agents, servants, employees, and all other persons in privity or acting in concert with Quantum from further infringement of the '249 Patent;

F. A judgment in favor of Riverbed denying Quantum all relief requested in its Complaint in this action and dismissing Quantum's Complaint for patent infringement with prejudice;

G. Judgment declaring that Riverbed has not infringed and is not infringing any claim of the '810 patent, and that Riverbed has not contributed to or induced and is not contributing to or inducing infringement of any claim of the '810 patent;

H. Judgment declaring that each claim of the '810 patent is invalid;

I. A declaration that this case is exceptional, and for an award to Riverbed of attorneys' fees, expenses, and costs pursuant to 35 U.S.C. § 285;

J. Disgorgement of all profits, revenues or investments that Quantum receives as a result of any wrongful action as averred herein by Riverbed; and

K. An award to Riverbed of such other and further relief as this Court deems just and proper.

DEMAND FOR JURY TRIAL

In accordance with Fed. R. Civ. P. 38(b), Riverbed demands a trial by jury on all issues raised in Riverbed's answer and counterclaims so triable.

DATED: November 13, 2007

QUINN EMANUEL URQUHART OLIVER & HEDGES, LLP

By /s/ Claude M. Stern
Claude M. Stern
Attorneys for Defendant and Counterclaimant,
RIVERBED TECHNOLOGY, INC.

EXHIBIT A

(12) **United States Patent**
McCanne et al.

(10) **Patent No.: US 7,116,249 B2**
(45) **Date of Patent: Oct. 3, 2006**

(54) **CONTENT-BASED SEGMENTATION SCHEME FOR DATA COMPRESSION IN STORAGE AND TRANSMISSION INCLUDING HIERARCHICAL SEGMENT REPRESENTATION**

(75) Inventors: **Steven McCanne**, Berkeley, CA (US); **Michael J. Demmer**, San Francisco, CA (US)

(73) Assignee: **NBT Technology, Inc.**, San Francisco, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **11/223,789**

(22) Filed: **Sep. 8, 2005**

(65) **Prior Publication Data**

US 2006/0061495 A1 Mar. 23, 2006

Related U.S. Application Data

(63) Continuation of application No. 10/968,868, filed on Oct. 18, 2004, now Pat. No. 6,961,009, which is a continuation of application No. 10/731,687, filed on Dec. 8, 2003, now Pat. No. 6,828,925, which is a continuation of application No. 10/285,330, filed on Oct. 30, 2002, now Pat. No. 6,667,700.

(51) **Int. Cl.**
H03M 7/38 (2006.01)

(52) **U.S. Cl.** **341/50**; 395/200.33; 715/744; 709/203

(58) **Field of Classification Search** 341/51, 341/50, 106; 715/744; 709/203; 395/200.33
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,377,803 A 3/1983 Lotspiech et al.
5,414,850 A 5/1995 Whiting
5,737,594 A 4/1998 Williams
5,822,746 A 10/1998 Williams
5,990,810 A 11/1999 Williams
6,163,811 A 12/2000 Porter
6,333,932 B1 12/2001 Kobayasi et al.

(Continued)

OTHER PUBLICATIONS

Chakrabarti; "Low-Bandwidth Web Access with Tandem Proxies;" Sep. 2002; 1-64; Massachusetts Institute of Technology.

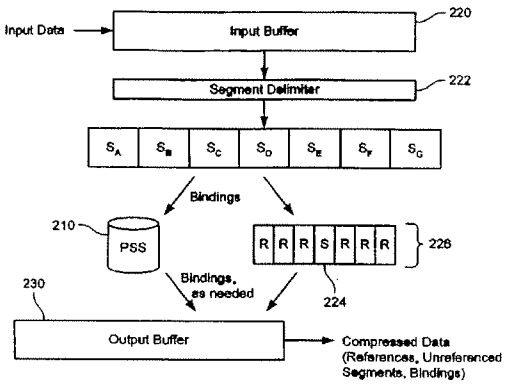
(Continued)

Primary Examiner—Peguy Jeanpierre
Assistant Examiner—Joseph Lauture
(74) *Attorney, Agent, or Firm*—Townsend and Townsend and Crew LLP

(57) **ABSTRACT**

In a coding system, input data within a system is encoded. The input data might include sequences of symbols that repeat in the input data or occur in other input data encoded in the system. The encoding includes determining a target segment size, determining a window size, identifying a fingerprint within a window of symbols at an offset in the input data, determining whether the offset is to be designated as a cut point and segmenting the input data as indicated by the set of cut points. For each segment so identified, the encoder determines whether the segment is to be a referenced segment or an unreferenced segment, replacing the segment data of each referenced segment with a reference label and storing a reference binding in a persistent segment store for each referenced segment, if needed. Hierarchically, the process can be repeated by grouping references into groups, replacing the grouped references with a group label, storing a binding between the grouped references and group label, if one is not already present, and repeating the process. The number of levels of hierarchy can be fixed in advanced or it can be determined from the content encoded.

30 Claims, 9 Drawing Sheets



U.S. PATENT DOCUMENTS

6,415,329	B1	7/2002	Gelman et al.
6,449,658	B1	9/2002	Lafe et al.
6,553,141	B1	4/2003	Huffman
6,642,860	B1	11/2003	Meulenbroeks
6,667,700	B1 *	12/2003	McCanne et al.
6,678,828	B1	1/2004	Pham et al.
6,791,947	B1	9/2004	Oskouy et al.
6,828,925	B1	12/2004	McCanne et al.
6,961,009	B1	11/2005	McCanne et al.
2002/0003795	A1	1/2002	Oskouy et al.
2002/0087547	A1	7/2002	Kausik et al.
2002/0138511	A1	9/2002	Psounis et al.
2002/0194382	A1	12/2002	Kausik et al.

OTHER PUBLICATIONS

Manber, Udi et al., "A Text Compression Scheme That Allows Fast Searching Directly in the Compressed File", *Department of Computer Science*, Mar. 1993, pp. 1-12, Technical Report #93-07, University of Arizona, Tucson, Arizona.

Manber, Udi et al., "Finding Similar Files in a Large File System", *Department of Computer Science*, Oct. 1993, pp. 1-10, Technical Report #93-33, University of Arizona, Tucson, Arizona.

Manber, Udi et al., "GLIMPSE: A Tool to Search Through Entire File Systems", *Department of Computer Science*, Oct. 1993, pp. 1-10, Technical Report #93-34, University of Arizona, Tucson, Arizona.

Mellia, Marco et al.; "TCP Smart-Framing: using smart segments to enhance the performance of TCP"; Dipartimento di Elettronica, Politecnico di Torino, 10129 Torino, Italy, pp. 1708-1712, date unknown.

Muthitacharoen, et al., "A Low-bandwidth Network File System", Symposium on Operating Systems Principles, 2001, pp. 147-187, URL=http://www.pdos.lcs.mit.edu/papers/lbfs:sosp01/lbfs.pdf, no month.

Sayood, Khalid et al., "Recursively Indexed Quantization of Memoryless Sources", *IEEE Transactions On Information Theory*, Sep. 1992, pp. 1602-1609, vol. IT-38, No. 5.

Spring, Neil T., "A Protocol-Independent Technique for Eliminating Redundant Network Traffic", Aug. 2000, Proceedings of {ACM} {SIGCOMM}, 9 pp., URL=http://www.acm.org/sigs/sigcomm//sigcomm2000/conf/paper/sigcomm2000-3-1.pdf.

"Unleashing the True Power of Today's Networks", *A Peribit White Paper*, Aug. 2002, pp. 1-13, URL=http://www.peribit.com/products/etc/0217w02punl.htm.

* cited by examiner

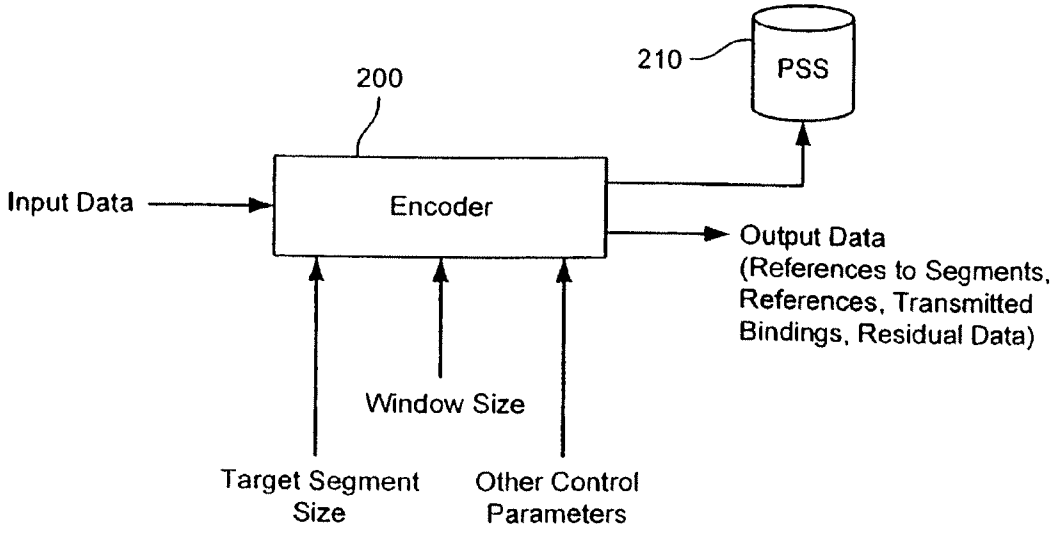

210
PSS
200
Input Data
Encoder
Output Data
(References to Segments,
References, Transmitted
Bindings, Residual Data)
Window Size
Target Segment
Size
Other Control
Parameters

FIG. 1

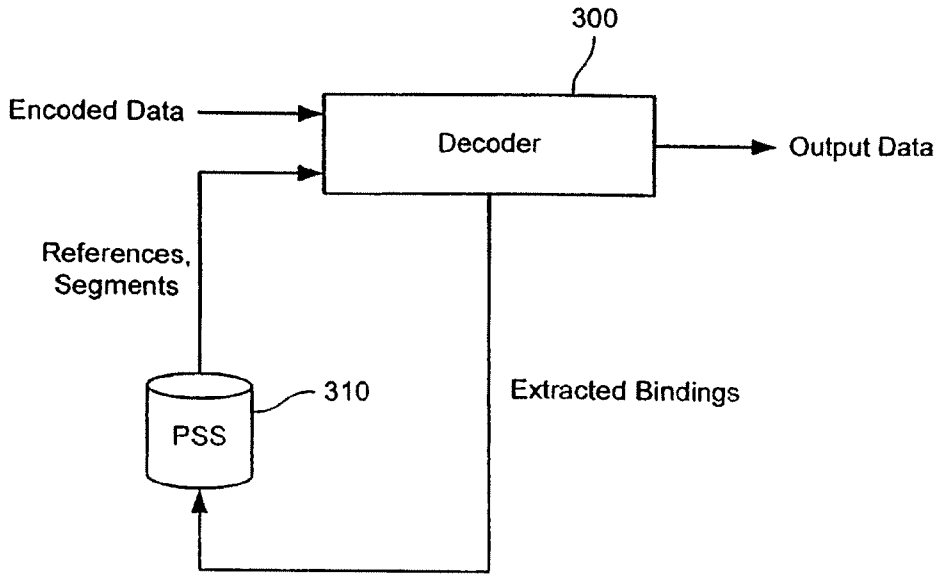

300
Encoded Data
Decoder
Output Data
References,
Segments
310
PSS
Extracted Bindings

FIG. 2

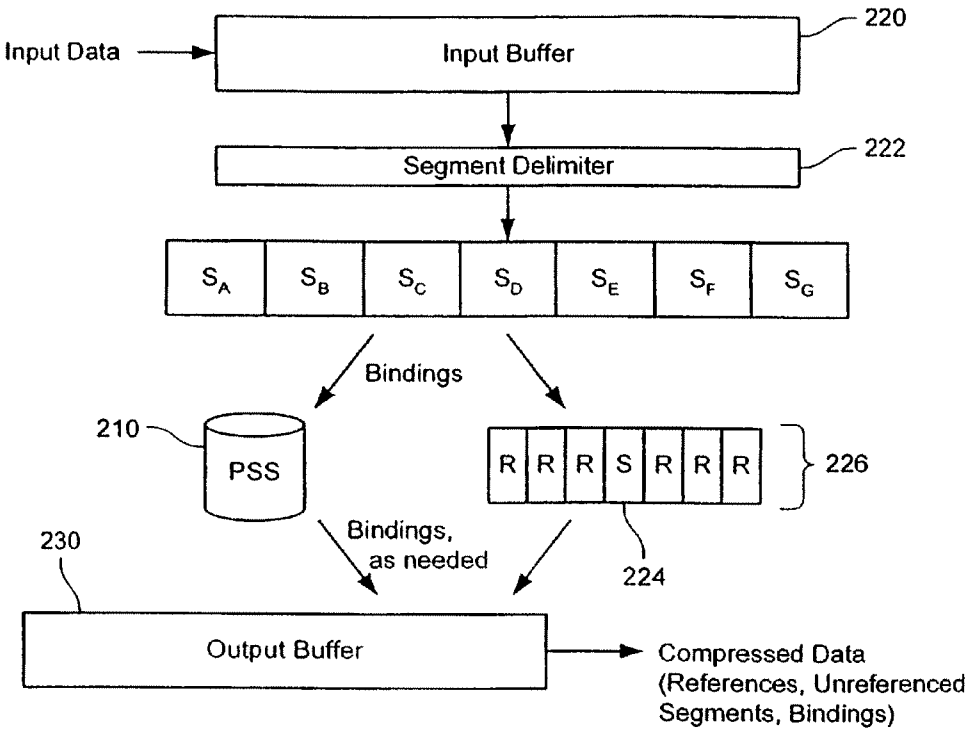

Input Data
Input Buffer
220
Segment Delimiter
222
S_A S_B S_C S_D S_E S_F S_G
Bindings
210
PSS
R R R S R R R
226
Bindings, as needed
224
230
Output Buffer
Compressed Data (References, Unreferenced Segments, Bindings)

FIG. 3

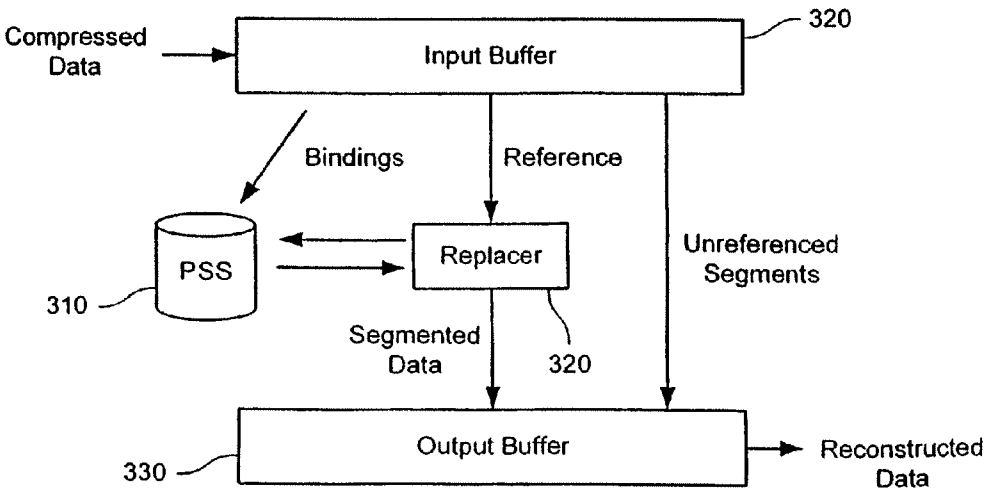

Compressed Data
Input Buffer
320
Bindings
Reference
PSS
310
Replacer
Unreferenced Segments
Segmented Data
320
Output Buffer
330
Reconstructed Data

FIG. 4

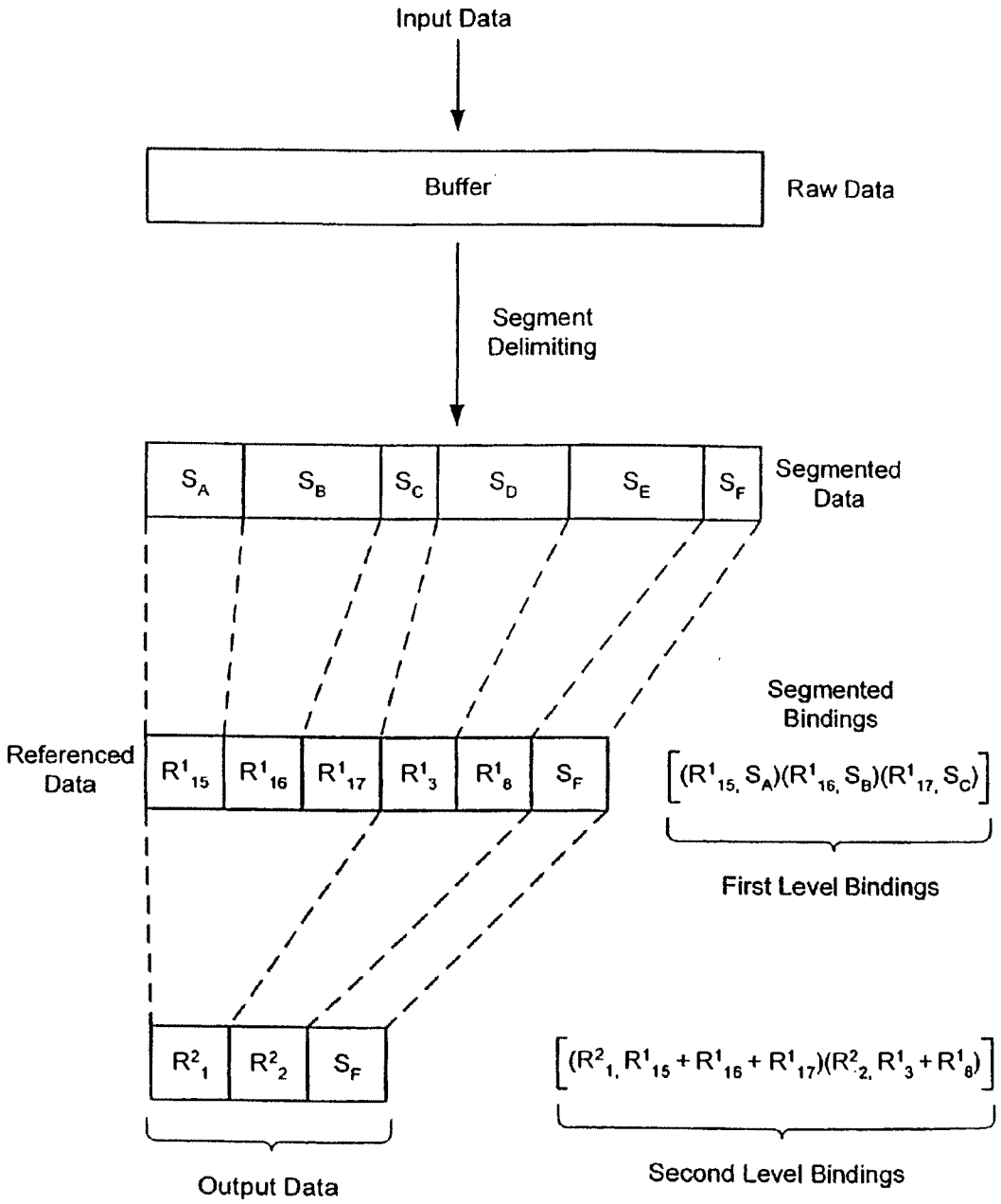

Input Data
Buffer
Raw Data
Segment Delimiting
S_A S_B S_C S_D S_E S_F
Segmented Data
Segmented Bindings
Referenced Data
R^1_{15} R^1_{16} R^1_{17} R^1_3 R^1_8 S_F
$[(R^1_{15}, S_A)(R^1_{16}, S_B)(R^1_{17}, S_C)]$
First Level Bindings
R^2_1 R^2_2 S_F
$[(R^2_1, R^1_{15} + R^1_{16} + R^1_{17})(R^2_2, R^1_3 + R^1_8)]$
Output Data
Second Level Bindings

FIG. 5

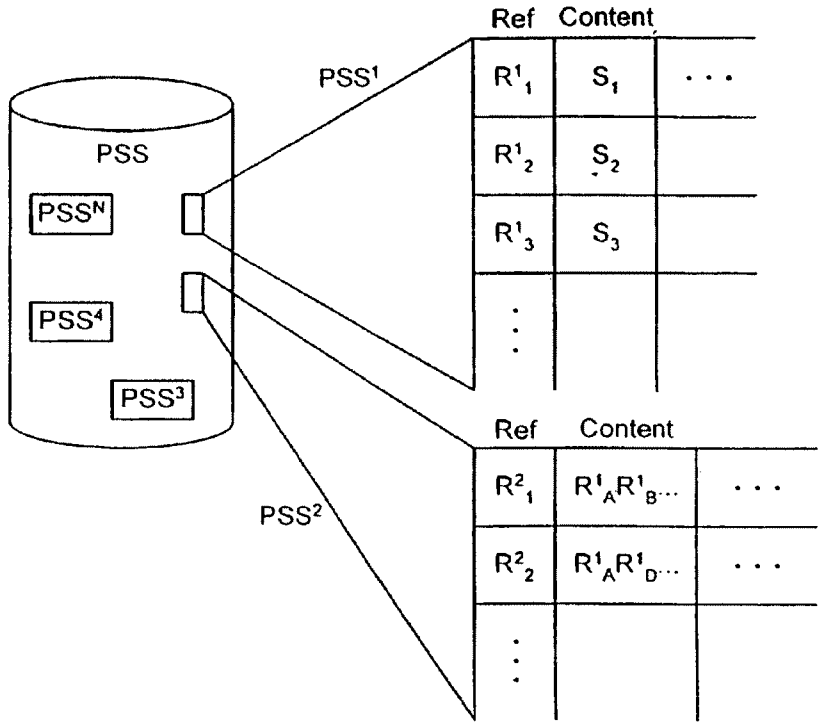

PSS
PSS^N
PSS^4
PSS^3
PSS^1
PSS^2
Ref Content
R^1_1 S_1
R^1_2 S_2
R^1_3 S_3
Ref Content
R^2_1 R^1_A R^1_B ...
R^2_2 R^1_A R^1_D ...

FIG. 6

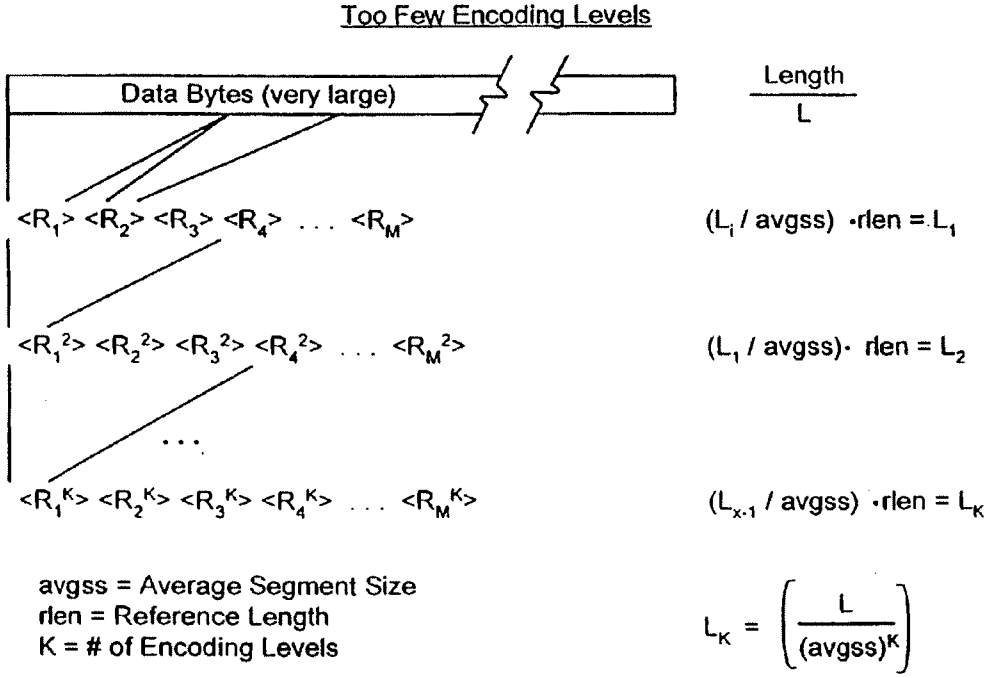

Too Few Encoding Levels
Data Bytes (very large)
Length
L
<R_1> <R_2> <R_3> <R_4> . . . <R_M>
(L_i / avgss) ·rlen = L_1
<R_1^2> <R_2^2> <R_3^2> <R_4^2> . . . <R_M^2>
(L_1 / avgss)· rlen = L_2
<R_1^K> <R_2^K> <R_3^K> <R_4^K> . . . <R_M^K>
(L_x-1 / avgss) ·rlen = L_K
avgss = Average Segment Size
rlen = Reference Length
K = # of Encoding Levels
L_K = (L / (avgss)^K)

FIG. 7

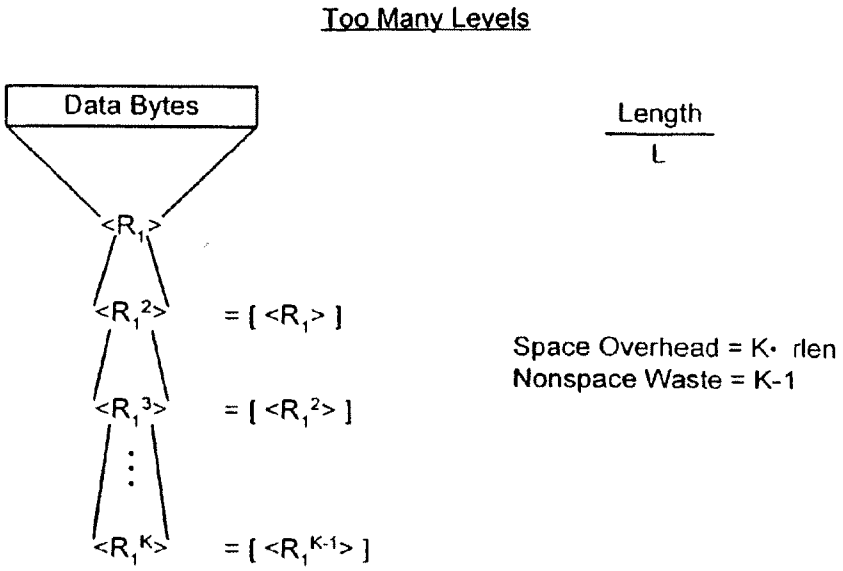

Too Many Levels
Data Bytes
Length
L
<R1>
<R1 2>
= [<R1>]
<R1 3>
= [<R1 2>]
<R1 K>
= [<R1 K-1>]
Space Overhead = K· rlen
Nonspace Waste = K-1

FIG. 8

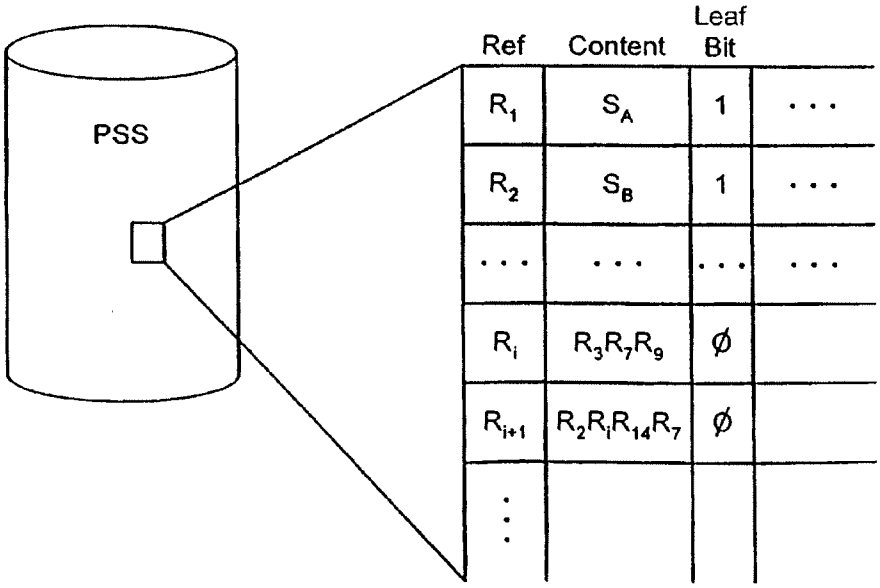

PSS
Ref
Content
Leaf Bit
R1
SA
1
R2
SB
1
Ri
R3R7R9
Ø
Ri+1
R2RiR14R7
Ø

FIG. 9

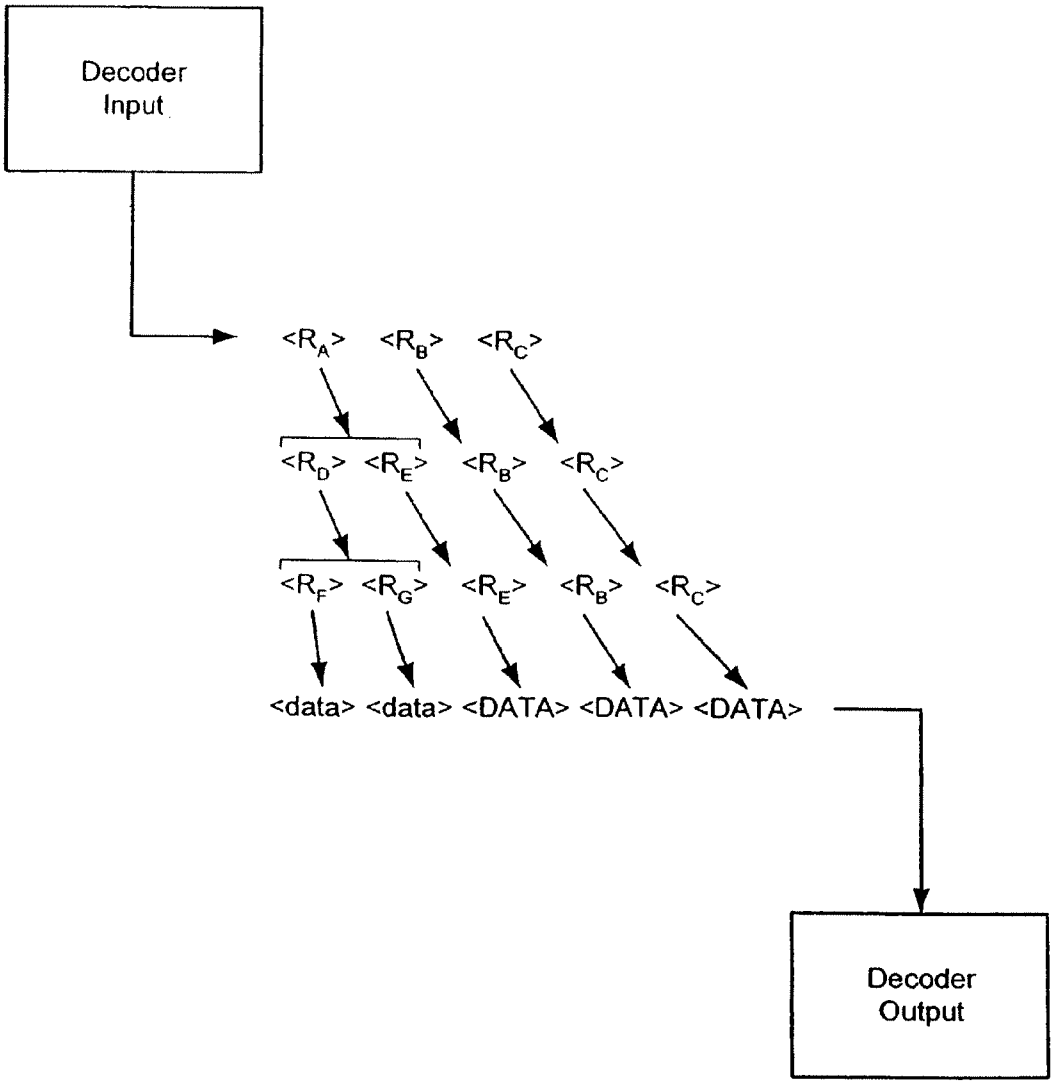

Decoder Input
<R_A> <R_B> <R_C>
<R_D> <R_E> <R_B> <R_C>
<R_F> <R_G> <R_E> <R_B> <R_C>
<data> <data> <DATA> <DATA> <DATA>
Decoder Output

FIG. 10

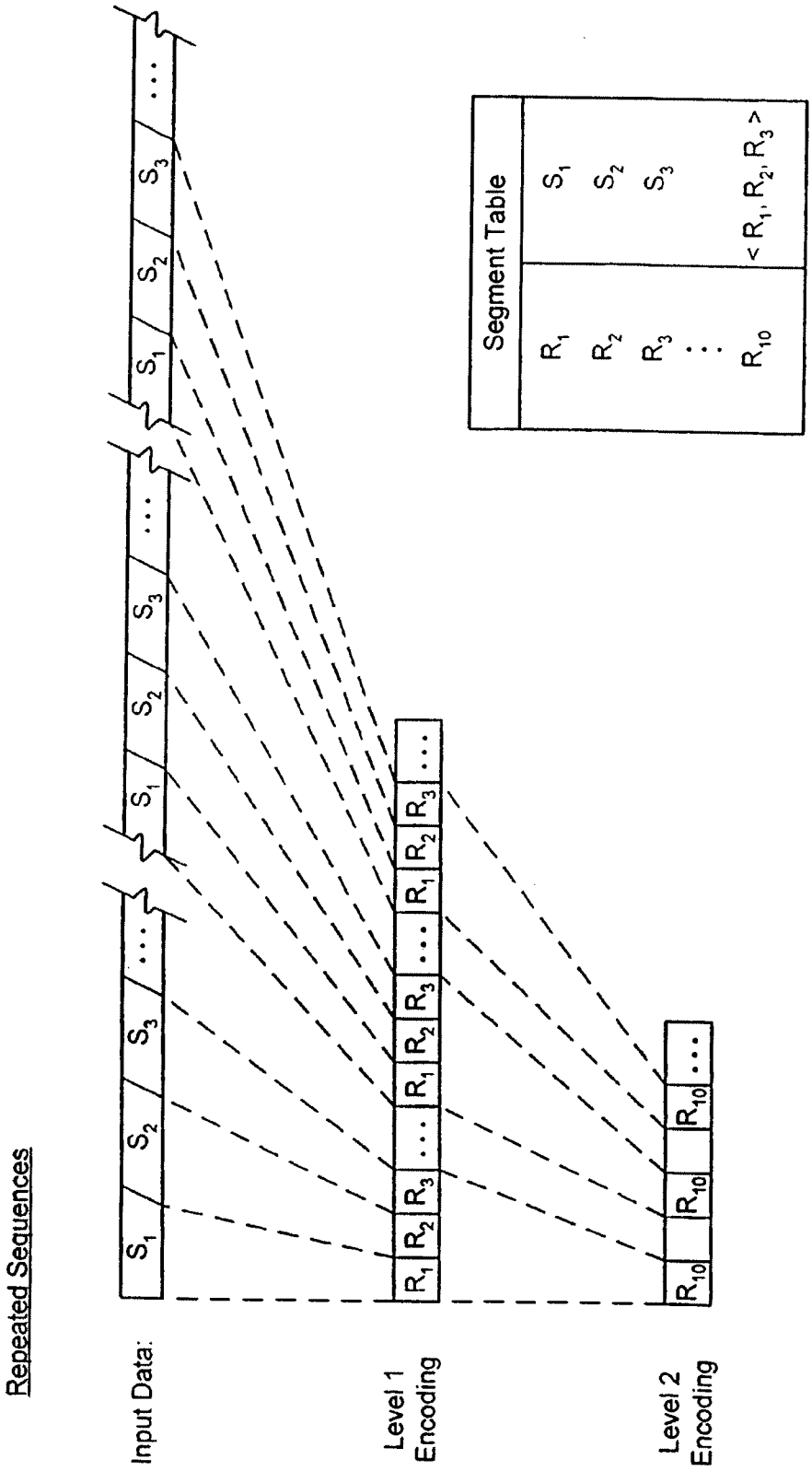

Repeated Sequences
Input Data:
Level 1 Encoding
Level 2 Encoding
Segment Table
S_1
S_2
S_3
$< R_1, R_2, R_3 >$
R_1
R_2
R_3
R_{10}

FIG. 11

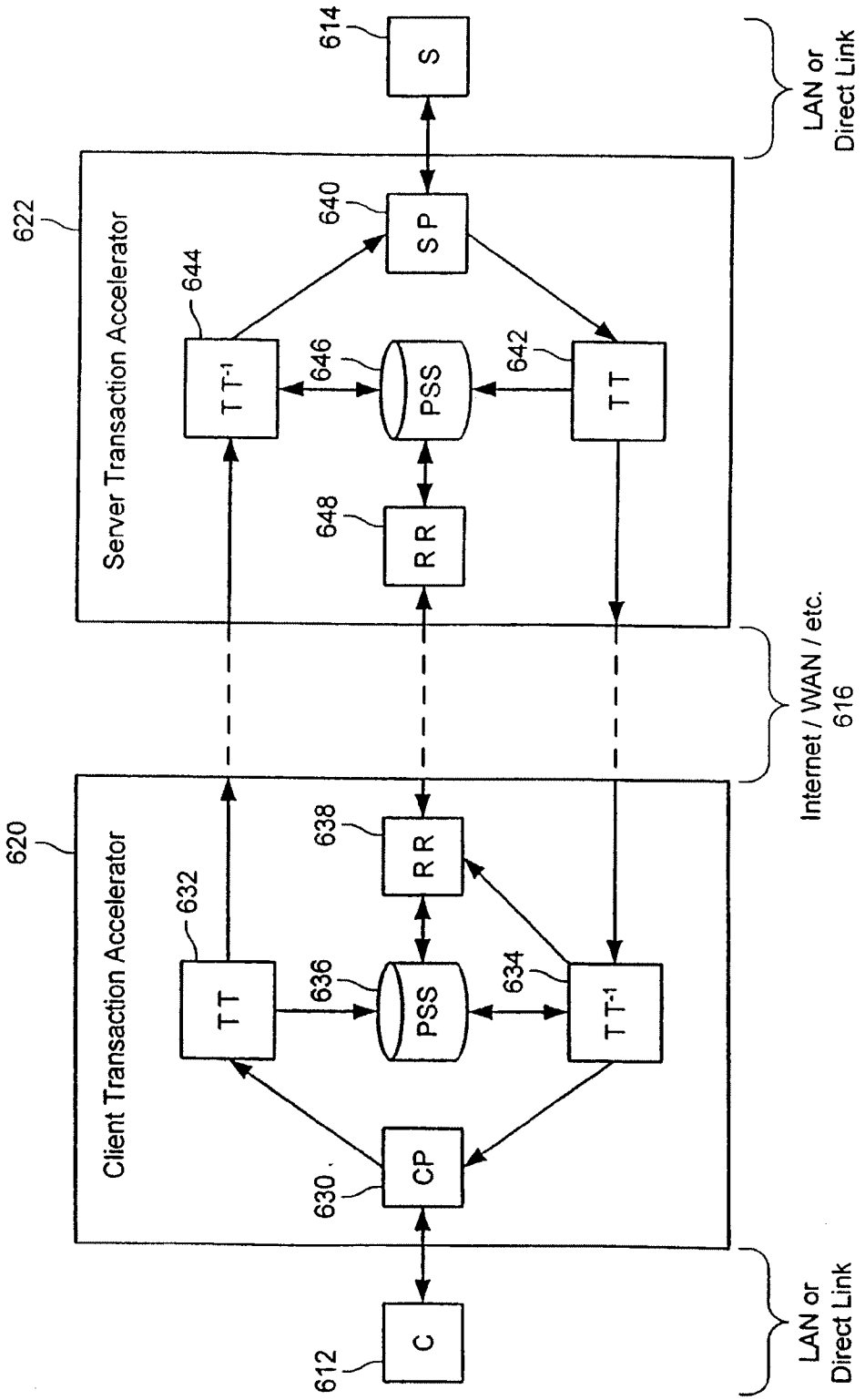

614
S
LAN or Direct Link
622
Server Transaction Accelerator
640
S P
644
T T-1
646
PSS
642
T T
648
R R
Internet / WAN / etc.
616
620
Client Transaction Accelerator
638
R R
632
T T
636
PSS
634
T T-1
630
CP
612
C
LAN or Direct Link

FIG. 12

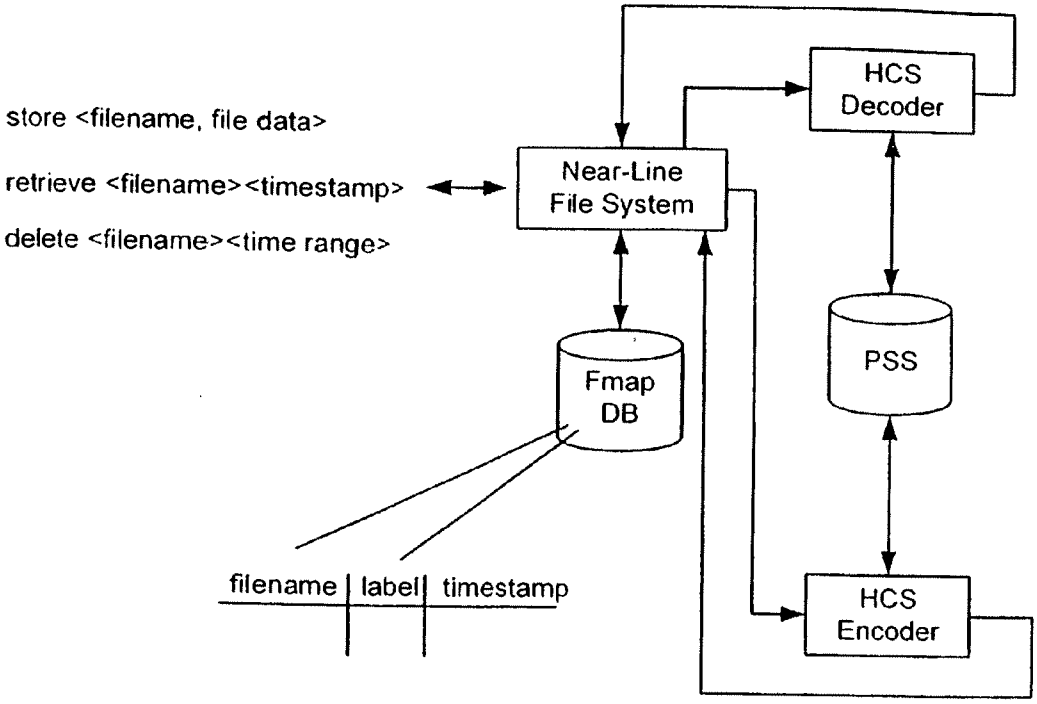

store <filename, file data>
retrieve <filename><timestamp>
delete <filename><time range>
HCS Decoder
Near-Line File System
PSS
Fmap DB
filename
label
timestamp
HCS Encoder

FIG. 13

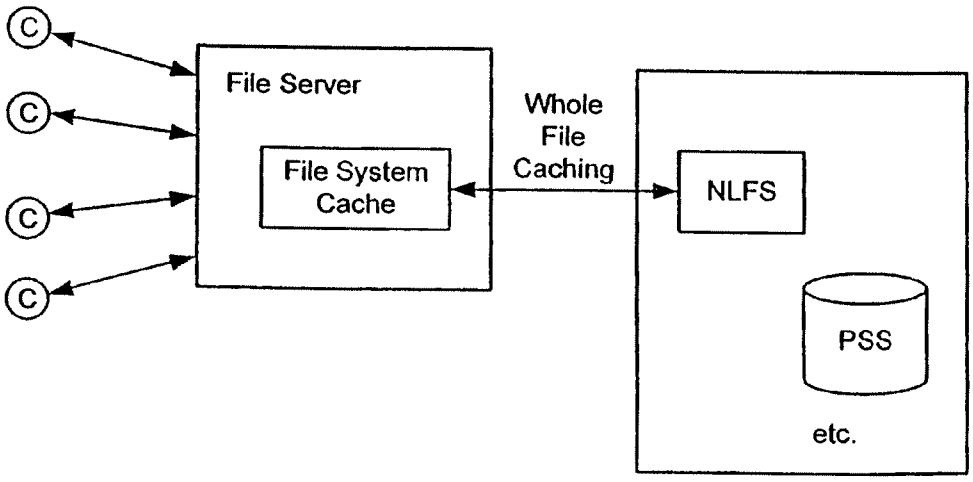

C
C
C
C
File Server
File System Cache
Whole File Caching
NLFS
PSS
etc.

FIG. 14

CONTENT-BASED SEGMENTATION SCHEME FOR DATA COMPRESSION IN STORAGE AND TRANSMISSION INCLUDING HIERARCHICAL SEGMENT REPRESENTATION

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a Continuation of U.S. patent application Ser. No. 10/968,868, filed Oct. 18, 2004 (now U.S. Pat. No. 6,961,009) entitled, CONTENT-BASED SEGMENTATION SCHEME FOR DATA COMPRESSION IN STORAGE AND TRANSMISSION INCLUDING HIERARCHICAL SEGMENT REPRESENTATION which is a Continuation of U.S. patent application Ser. No. 10/731,687, filed Dec. 8, 2003 (now U.S. Pat. No. 6,828,925) entitled, CONTENT-BASED SEGMENTATION SCHEME FOR DATA COMPRESSION IN STORAGE AND TRANSMISSION INCLUDING HIERARCHICAL SEGMENT REPRESENTATION which is a Continuation of U.S. patent application Ser. No. 10/285,330, filed Oct. 30, 2002 (now U.S. Pat. No. 6,667,700) entitled CONTENT-BASED SEGMENTATION SCHEME FOR DATA COMPRESSION IN STORAGE AND TRANSMISSION INCLUDING HIERARCHICAL SEGMENT REPRESENTATION all of which are hereby incorporated by reference, as if set forth in full in this document, for all purposes.

This application is related to co-pending U.S. patent application Ser. No. 10/285,315 entitled TRANSACTION ACCELERATOR FOR CLIENT-SERVER COMMUNICATION SYSTEMS filed Oct. 20, 2002 and is hereby incorporated by reference, as if set forth in full in this document, for all purposes.

BACKGROUND OF THE INVENTION

The present invention relates generally to data compression and more specifically to segmentation used for compression.

Data compression is useful for more efficiently storing and transmitting data. Data compression is a process of representing input data as compressed data such that the compressed data comprises fewer bits or symbols than the input data and is such that the compressed data can be decompressed into at least a suitable approximation of the original input data. Compression allows for more efficient transmission of data, as fewer bits need to be sent to allow a receiver to recover the original set of bits (exactly or approximately) and compression allows for more efficient storage as fewer bits need be stored.

"Compression ratio" refers to the ratio of the number of bits or symbols in the original data to the number of bits or symbols in the compressed data. For example, if a sequence of 100 bytes of data is representable by 5 bytes of data, the compression ratio in that example is 20:1. If the input data need not be recovered exactly, so called "lossy compression" can be used, generally resulting in greater compression ratios than "lossless" compression. In a typical application where the compression is to be transparent, the compression should be lossless.

Compression based on the structure and statistics of the input content is common. A typical compressor receives an input stream or block of data and produces a compressed stream or block, taking into account the symbol values in the input, the position of particular symbol values in the input, relationships among various symbol values in the input, as

well as the expected nature of the source of input data. For example, where the input data is expected to be English text, it is highly likely that the output of the source following a "." (period) symbol is a " " (blank space) symbol. This characteristic of the source can be exploited by the compressor. For example, the blank space might be represented by no symbol at all in the compressed data, thus reducing the data by one symbol. Of course, in order to have the compressed data be decompressable losslessly, the compressor would have to encode special notations for each instance where a period is not followed by a blank space. However, given their relative frequency of occurrence, many more omissions can be expected than special notations, so the overall result is net compression.

One method of compression used with sources that are likely to contain repeated sequences of input characters is the dictionary approach. With this approach, a dictionary of symbol sequences is built up and each occurrence of one of the symbol sequences in the dictionary is replaced with the index into the dictionary. Where the compressor and the decompressor have access to the same dictionary, the decompressor can losslessly decompress the compressed data by replacing each dictionary reference with the corresponding entry. Generally, dictionary compression assumes that an input stream can be divided into sequences and that those sequences will recur later in the input stream.

Of course, for the dictionary approach to work, the decompressor has to have a copy of the dictionary used by the compressor. Where the compression is for reducing transmission efforts, the compressor and the decompressor are normally separated by the transmission channel over which efforts are being reduced, but the load on the channel may be increased if the dictionary is sent over that channel. A similar issue arises where compression is to be applied for reducing storage, as the dictionary needs to be stored so the decompressor has access to it and that adds to the storage effort. In some schemes, the dictionary is a fixed dictionary and thus it can be amortized over many compressions to reduce the per compression cost of the dictionary to where the overhead is insignificant. In other schemes, the dictionary is adaptive, but is reconstructable from data already available to the decompressor, but as previously decompressed symbols.

Compression is useful in networks where network traffic is limited by bandwidth constraints. One example is a wide area network (WAN), such as the Internet, which generally has less free bandwidth per use than other networks, such as a dedicated local area network (LAN) or a dedicated WAN. For cost reasons, many would like to use nondedicated WANs instead of relying only on LANs or adding dedicated WANs, but are constrained by the performance of nondedicated WANs. Compression can potentially make it feasible to use a low bandwidth link for high bandwidth applications since it reduces the number of actual bits required to represent a larger input sequence. Similarly, compression can potentially enhance performance or capacity of a file system by reducing the number of bits required to represent all of the files in the system.

In general, data stored and communicated across enterprise systems and networks often has high degrees of information redundancy present. For example, e-mail messages and attachments sent to large numbers of recipients in a corporation generate many redundant copies of the message data in storage systems as well as cause redundant traffic to be sent across the network. Likewise, many electronic documents within an enterprise share very high

degrees of commonality as different employees work with similar pieces of corporate information in different settings.

If such data were compressed, network performance would improve and effective storage capacity would increase. Traditional compression schemes can exploit some of these redundancies by detecting statistical correlations in an input symbol stream and encoding the stream's symbols in as few bits as possible based on the statistical correlations. Some dictionary-based compression schemes are known as "universal codes" in that they converge to the optimal compression scheme (the Shannon limit) under various assumptions including the assumption that the input symbols conform to a stationary random process. This would imply then that one could achieve optimal performance simply by deploying a universal coding system that performed optimal compression of network traffic in a network or of file data in a storage system.

However, this approach does not necessarily work well in practice. For example, it is well known that enabling compression on the network interface of a router improves performance, but only marginally (30% is typical but it depends on the underlying traffic). One problem with traditional universal coding schemes is that they do not necessarily converge to optimal rate if the underlying data input has nonstationary statistics. Moreover, if the underlying statistics are stationary but they exhibit "long-range dependence" (LRD), the rate of convergence of the universal code to optimality could be impractically slow (perhaps exponentially slow). This has important consequences as many studies have provided evidence that network traffic exhibits LRD, and in fact, there is an open controversy as to whether the underlying data processes are best modeled as LRD random processes or non-stationary processes. Other studies have shown that file statistics (like size distributions, etc.) also exhibit LRD. In short, this all means that traditional methods of universal coding are not necessarily the best practical solution, and a technique that exploits long-range dependence of typical data sources is likely to do better.

One brute-force approach to detecting long-range correlations is to employ a dictionary-based compression scheme that searches with great breadth over a data source (a file, a communication stream, etc.) for patterns that are repeated, represent those patterns with a name or label and store the corresponding data in a table or database in association with the name or label. To exploit LRD, a very large window of data could be kept that allows the system to peer arbitrarily far back in the input (or in time) to detect long-range dependent patterns. This simple model intuitively matches the structure of information in an enterprise. That is, many similar sources of information both change slowly over time and appear in different contexts (email, file systems, Web, etc). As underlying technology improves (e.g., disks and memory become increasingly less expensive), this approach becomes even more practical. However, the brute-force approach still has shortcomings.

One shortcoming is that searching for arbitrary patterns of matching data in a bit stream is computationally expensive and the general problem of finding the optimal solution quickly and efficiently in the presence of LRD statistics has not been adequately solved. An alternative approach is to abandon the ideal of finding an optimal solution and instead focus on approximate solutions or heuristics that perform well in the light of LRD and are practical and feasible.

One tool that proves useful in this framework is a proposed heuristic for finding repeated patterns in data by segmenting the data based on the input content itself, rather than some externally imposed blocking or framing scheme. See, for example, Muthitacharoen, A., et al., "A Low-Bandwidth Network File System", in Proceedings of the 18th ACM Symposium on Operating Systems Principles, SOSP '01, Chateau Lake Louise, Banff, Canada, October 2001, in vol. 35, 5 of *ACM SIGOPS Operating Systems Review*, pp. 174–187, (ACM Press, New York, N.Y.), 2001. In the LBFS system described therein, portions of transmitted files are replaced with hashes, and the recipient uses the hashes to reconstruct which portion of which file-on-a-file system corresponds to the replaced data. Another example of segmentation based-on-input content is described in the context of matching portions of files, as described by Manber, "Finding Similar Files in a Large File System", *USENIX Proceedings*, San Francisco 1994 (available as *University of Arizona Dept. of Comp. Sci. Technical Report TR*93-33).

Other attempts to reduce network traffic through dictionary-style compression techniques have been applied at the network layer. One such technique includes representing portions of network traffic with tokens and maintaining tables of tokens at each end of a connection. See, for example, Spring, N., et al., "A Protocol-Independent Technique for Eliminating Redundant Network Traffic", in *Proceedings of ACM SIGCOMM* (August 2000). As described in that reference, network traffic that contains redundancies can be reduced by identifying repeated strings and replacing the repeated strings with tokens to be resolved from a shared table at either end of a connection. Because this approach operates solely on individual packets, the performance gains that accrue are limited by the ratio of the packet payload size to the packet header (since the packet header is generally not compressible using the described technique). Also, because the mechanism is implemented at the packet level, it applies only to regions of the network where two ends of a communicating path have been configured with the device. This configuration can be difficult to achieve, if not impractical, in certain environments. Also, by indexing network packets using a relatively small memory-based table with a first-in first-out replacement policy (without the aid of, for instance, a large disk-based backing store), the efficacy of the approach is limited to detecting and exploiting communication redundancies that are fairly localized in time, i.e., the approach can not exploit LRD properties of the underlying data stream.

An alternative approach to reduce network traffic involves caching, where a request for data is not sent over the network if a copy of the data is available locally in a cache. As used herein, the terms "near", "far", "local" and "remote" might refer to physical distance, but more typically they refer to effective distance. The effective distance between two computers, computing devices, servers, clients, peripherals, etc. is, at least approximately, a measure of the difficulty of getting data between the two computers.

While caching is good for blocks of data that do not change and are not found in similar forms under different names, improvements are still needed in many cases. In file caching, the unit of caching is typically a block of a file or the whole file. If the same data is present in a different file, or two files have only small differences, caching will not remove the redundancies or exploit them to reduce communication costs. Even if a data object is segmented into many blocks and each of the blocks is cached separately, the net result is still inefficient because a small insertion or deletion of data in the underlying object will cause the data to shift through many (if not all) of the blocks and thus nullify the benefits of caching. This because the blocks are imposed

arbitrarily on the input stream, and so it is impossible to detect that only a small change has been made to the underlying data.

In view of the above, improvements can be made in compressing data in a network environment, in storage systems, and elsewhere.

BRIEF SUMMARY OF THE INVENTION

In a coding system according to one embodiment of the present invention, input data within a system is encoded. The input data might include sequences of symbols that repeat in the input data or occur in other input data encoded in the system. The encoding includes determining one or more target segment sizes, determining one or more window sizes, identifying a fingerprint within a window of symbols at an offset in the input data, determining whether the offset is to be designated as a cut point and segmenting the input data as indicated by the set of cut points. For each segment so identified, the encoder determines whether the segment is to be a referenced segment or an unreferenced segment, replacing the segment data of each referenced segment with a reference label and storing a reference binding in a persistent segment store for each referenced segment, if needed. Hierarchically, the process can be repeated by segmenting the reference label strings into groups, replacing the grouped references with a group label, storing a binding between the grouped references and group label, if one is not already present, and repeating the process. The number of levels of hierarchy can be fixed in advanced or it can be determined from the content encoded.

Other features and advantages of the invention will be apparent in view of the following detailed description and preferred embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of an encoder that encodes a data stream or block using segmentation based on content and segment references.

FIG. **2** is a block diagram of a decoder that decodes a data stream or block using segmentation bindings from a persistent segment store.

FIG. **3** is a diagram illustrating an encoding process in more detail.

FIG. **4** is a diagram illustrating a decoding process in more detail.

FIG. **5** is a diagram illustrating a hierarchical encoding process.

FIG. **6** is an illustration of a persistent segment store having a plurality of level segment stores.

FIG. **7** diagrammatically illustrates a problem with using too few encoding levels.

FIG. **8** diagrammatically illustrates a problem with using too few encoding levels.

FIG. **9** is an illustration of a persistent segment store organized to hold an arbitrary depth of references.

FIG. **10** illustrates a decoding process using the persistent segment store of FIG. **9**.

FIG. **11** illustrates hierarchical content-induced segmentation.

FIG. **12** is a block diagram of a networked client-server pair where some traffic between the client and the server is routed through a client-side transaction accelerator ("CTA") and a server-side transaction accelerator ("STA").

FIG. **13** illustrates a file system using hierarchical content-induced segmentation.

FIG. **14** illustrates a near-line file system (NLFS) and file server front end.

DETAILED DESCRIPTION OF THE INVENTION

The present invention has many applications, as will be apparent after reading this disclosure. In describing an embodiment of a compression system according to the present invention, only a few of the possible variations are described. Other applications and variations will be apparent to one of ordinary skill in the art, so the invention should not be construed as narrowly as the examples, but rather in accordance with the appended claims.

Coding (encoding, decoding) can be done by a number of different devices, which might involve hardware, software, or both. Coding can be done with a computer, computing device, peripheral, electronics, or the like, and/or using an application being executed or controlled by such element. Coding might be done incident to a transport process, such as that described in McCanne I. Using the coding apparatus and processes described herein, the responsiveness of transactions over a network can be improved by lessening the number of bits that need to be transmitted at critical times over critical channels. Also, the coding system may be integrated into a stand-alone storage system to optimize capacity. In this environment, the effective capacity of a storage system can be enhanced as the coding system described herein extracts common sequences of data and stores them just once, even if the sequence appears potentially many times, in potentially many different files.

FIG. **1** illustrates the particular inputs that an encoder might have. As shown there, an encoder has an input for receiving input data and inputs for parameters, such as target segment size, window size and other control parameters, as well as outputs for the output data and bindings generated in the encoding process, which are stored in a persistent segment store (PSS) **210** as needed. In operation, encoder **140** would process input data, identify segments of data, replace the segment's data with a reference, provide the segment data and a segment reference to PSS **142** in the form of a binding and output the encoded data. The encoded output data might include unreferenced segment data, embedded bindings and/or reference labels for referenced segments. In the simplest case, the output data is entirely reference labels.

The target segment size parameter is used by the encoder to control the average size of a segment. In general, segments are variable length and there are design tradeoffs in selection of this size as described below.

FIG. **2** illustrates a decoder **300** and a PSS **310**, which together perform decoding that is the inverse of the encoding done by encoder **200**. As described above, the encoded data might comprise references, bindings and unreferenced residual data. When decoder **150** encounters a binding in received data, it can use the segment data in that binding to reconstruct the original data and it can also store the binding in its PSS. When decoder **150** encounters a reference without a binding, it can use the reference to obtain segment data from PSS **152** to reconstruct the segment. If the segment reference is not found in PSS **152**, decoder **150** can send a request for the segment data.

FIG. **3** is a diagram illustrating an encoding process in more detail. As shown there, input data to be encoded is stored in buffer **220**, segmented by segment delimiter **222**, which creates the output references and bindings for PSS **210**. The output references, unreferenced segments, such as

segment **224**, and bindings as needed are provided to an output buffer **230** to form the output of the encoder.

FIG. **4** is a diagram illustrating a decoding process in more detail. As shown there, encoded data is buffered in an input buffer **320**. From the buffer contents, bindings are extracted and stored in PSS **310**. Reference labels are provided to a replacer **325** that replaces them with the referenced segment data and places the data in an output buffer **330**. Unreferenced segment data is output directly to output buffer **330** for eventual output as the reconstructed data.

As the above-described figures illustrate, one aspect of the encoding process is the segmentation of input data. In a process for segmenting, identifying "cut points", such as offsets in the input data where one segment ends and the next segment begins, is equivalent to segregating the input data into separate data structures, or the like.

If a segmentation scheme is designed appropriately, segment boundaries should always appear in the same place for the same sequences of data, regardless of the context in which that data appeared. If that were the case, commonly repeated patterns of data would be segmented in the same way over each repetition of data and a system could be designed to efficiently identify these repeated patterns. For example, a particular data sequence (such as pieces of a widely used GIF image or pieces of a bitmap representing a commonly used graphical icon) that appears in many different locations in larger files (such as a word processing document, slide presentation document, or on a Web page) will always be found and segmented in the same way, regardless of what data surrounds it.

To achieve this property, the segmentation scheme herein uses information in the data itself to guide the segmentation process rather than externally imposed parameters like block sizes, transaction boundaries, etc. As input data is consumed by the coding process, the various values and structure of the input symbols guide the segmentation process (as described below). Consequently, the system has a "self-synchronizing" property where if similar data patterns are presented to its input, the same segment boundaries are detected, inducing the formation of the same segments that have been seen in the past. Moreover, this means that the system is robust to insertions, deletions, or other changes in the underlying input data, in that once a previously present segment boundary is found, the new segment will match an existing segment from that point on (assuming the data pattern from that point on has been seen in the past).

Because this scheme is guided by the content of the input, it is described herein as "content-induced segmentation". By applying content-induced segmentation to an input stream with repeated patterns across very large time scales (i.e., exhibiting LRD statistics), the same segments are created without having to keep track of the entire history of data patterns that have been seen in the past. That is, the segmentation process can simply segment the input based on the input itself without having to search over the existing data patterns already found. While, in general, this approach does not produce the optimal segmentation (i.e., maximizing the size of segments while simultaneously maximizing the number of repeated segments found), it provides a good tradeoff between complexity and performance—the scheme is effective at exploiting certain types of LRD while leading to an efficient and practical implementation.

Content-induced Segmentation

A segmenter, such as segment delimiter **222** in FIG. **3** operates on a portion (or all) of a set of input data. The segmenter batches up a certain number of input symbols (determined by the current offset and the window size parameter) and computes a hash function over the window. This is referred to as the fingerprint of the window. A deterministic fingerprint indicator function returns a Boolean value for each fingerprint indicating whether the offset is to be considered a cut point and thereby defines a segment boundary. Preferably, the window does not extend beyond either end of an application data unit (ADU), so that each window contribution is from exactly one ADU. Thus, preferably, the encoder is provided an indication in an input data stream where one ADU ends and another one starts. However, such indication is not necessary for correct operation of the system. Rather, these indications allow the encoder to force a segment boundary in the stream where an ADU boundary ends and thus prevent practical inefficiencies such as extra delays waiting for data that might not ever arrive (e.g., relying upon an implementation time-out to force a segment boundary instead). Additionally, restricting segment boundaries to be within a single ADU prevents the system from detecting and storing segments that are unlikely to be repeated in the future as may happen if a segment spans two ADUs.

Where the fingerprint indicator function values are 1 and 0, 1 might represent a cut point and 0 represent other than a cut point. Thus, under one convention, when the function evaluates to 1 for a given fingerprint having a given offset and window, a new segment is created at the symbol defined by the first symbol of the input data (in order according to an input data order). Under that convention, if the function evaluates to 0, then one more input symbol is consumed and the window is advanced to cover this new symbol (and the least current symbol is removed from the window but remains in the current segment). The target segment size can thus be controlled as a parameter of the fingerprint indicator function.

Since a well-chosen fingerprint function will tend to create random bit patterns in the fingerprint, the Bernoulli distribution function of a random variable, parameterized by the target segment size, can be used to map the hash value into the indicator value. For example, the parameter can be chosen such that on average, assuming a random distribution of fingerprint inputs, 1 out of M times, the function evaluates to true. Thus, on average, the segment size is going to be M+W bytes, where W is the window size in use (because a segment is at least W bytes large). Thus, the value of M determines the target segment size.

A variation of the segmentation process allows the window to define the start of a segment boundary instead of the end of a boundary. This allows segments to be arbitrarily small, since the pattern of data used to define the fingerprint window is not necessarily consumed as part of the segment.

In another embodiment, an explicit fingerprint window is not needed. Instead, a hash function is computed over a variable number of input symbols. As each input symbol is consumed, the fingerprint function is re-computed over the entire buffer of consumed symbols. Again, a subsequent indicator function applied to the fingerprint decides when and if to insert a new segment boundary at the current point in the input. This approach is, in general, less efficient than the previous approach because changes in data near the start of a segment impact the segmentation process going forward. Consequently, segment boundaries that would have otherwise been created and matched against previous segments are missed due to changes in data that can be far

removed from the desired segment boundary. The windowing approach described earlier accomplishes exactly this goal.

The fingerprint hashing scheme can be made efficient by using a hash function that can be incrementally updated as input symbols are read. Examples of efficient generation of hash functions for incrementally updated windows are known and need not be described in further detail here. See, for example, Rabin, "*Fingerprinting by Random Polynomials*", Technical Report TR-15-81, Dept. of Comp Sci., Harvard University (1981).

When a new segment is defined (by finding a new segment boundary), the segment is compared against all the existing segments that are stored in the PSS. This lookup process can be made efficient by maintaining an index of segments keyed by a hash function computed over the segment data (either the entire segment content, portions of the segment content, the segment fingerprint value, or combinations thereof). For each hash value, the index contains the set of segments present in the PSS that hash to that value. Thus, to determine if a segment exists in the PSS, the encoder computes the hash for the segment in question and performs a lookup on the segment index using said hash. If the lookup fails, the segment cannot be in the PSS. If the lookup succeeds, the encoder can compare each segment returned by the lookup to the segment in question to check for an exact match. This handles the rare case that multiple segments hash to the same hash index.

Continuing the description of the encoding process, if an acceptable segment (usually an identical segment, but in special cases, complete identity is not required) is not present, a new unique name is assigned to the new segment and the binding (reference label, segment data) is entered into the PSS. Where needed, the binding can be compressed, but in the simplest case where the PSS is not constrained, the binding might be stored as a record in a database with fields in each record for storing a string representing the reference label and data representing the segment data. In addition, the segment data within the binding could be itself compressed using a traditional compression algorithm that effectively exploits the short-range dependence present in any segment data. This could be especially useful in the nonterminal segments (described below) that comprise strings of labels that could have a high degree of redundancy depending on the details of the segment naming scheme.

If a comparable segment is present, then its previously defined reference label is used for the new segment and no new binding is created. That reference label is then output instead of the actual segment data, as illustrated by a sequence **226** shown in FIG. **3** Thus, the original input data is represented as a sequence of reference labels bound to data objects stored in the PSS. Some of the data might not be replaced by reference labels, such as where it is determined that replacement of a particular segment (for example, segment **224** in FIG. **3** with a reference label will not substantially improve performance. However, in some embodiments, all of the segments are represented with reference labels.

The reference labels might not be compactly represented, resulting in more bits being used for sequence **226** than necessary. If that is the case, reference labels might, in turn, be compressed using traditional methods for compression, e.g., differential encoding of the reference labels followed by run-length coding and or Huffman coding, or other approaches.

FIG. **4** illustrates a decoding process, which is the inverse of what is shown in FIG. **3** The encoded data (reference labels and possibly also bindings and unreferenced segment data) is received into input buffer **320**. From that input buffer data, bindings are extracted and stored in PSS **310** and unreferenced segment data is moved to output buffer **330**. Referenced segments, which are represented in the encoded data as reference labels (or compressed reference labels), are replaced with their segment data by replacer **325** using the reference labels to locate bindings in PSS **310** and thus obtain the corresponding segment data. Since reference labels are unique for unique data, the correct segment data can always be provided. If the referenced bindings are not present in PSS **310**, they can be requested from, for example, PSS **210**. Where encoding is done for storage purposes, PSS **210** and PSS **310** might be one data structure, written by the encoder and read by the decoder.

In many cases, the parameters for the encoding process need to be chosen carefully for good performance. If the target segment size chosen is very large, then the effective compression ratio might be high, because large numbers of symbols are grouped into segments and replaced with a relatively small number of bits representing the segment's reference label. However, a large target block size might tend to miss finer-grained repetitions of data, and add to the overhead of storing and moving multiple versions of nearly similar segment data. On the other hand, if the target segment size chosen is very small, the compression ratio might be low, because the number of bits required to represent a reference label might not be substantially less than the number of bits in the original segment data. In general, the degree to which a particular segmentation scheme is effective depends on the underlying statistics of the data. This problem can be solved by introducing hierarchy into the reference label scheme. This approach is referred to herein as Hierarchical Content-Induced Segmentation (HCS).

Hierarchical Content-Induced Segmentation

In order to get the benefits of large target block sizes (such as high compression ratios) and the benefits of small target block sizes (such as having finer-grained repetitions noticed and segmented as such), hierarchical referencing can be used. In such a system, input data to be encoded might be segmented using a small target block size, yielding many reference labels. The reference labels are, in turn, grouped and replaced with group labels, with reference bindings (group labels, sequence of reference labels forming the group) stored in a PSS-like structure. This allows a single technique to be used over various types of repetitive data, whether the repetitive patterns occur on a fine-grain or a course-grain basis. To capture fine-grained repetitive patterns, the target block size chosen is relatively small. A small target block size can be expected to result in a more verbose encoded reference stream, but the hierarchical referencing will tend to reduce the overhead of having many reference label patterns repeat, in effect having each resulting label in the final result represent as large a span of the input data as can be found repeating. The hierarchical referencing can use a different target block size at each level that is tuned to the relative size of the reference names at that level, or it can use the same size. Similarly it might use a different fingerprint function and/or a different fingerprint window size at each level in the hierarchy, or use the same functions uniformly throughout.

Two types of such schemes are described below. An example of hierarchical reference encoding is shown in FIG. **5**. An input buffer is loaded with input data to be encoded and that input data is segmented into segments S_A, S_B, S_C,

S_D, S_E and S_F. In this example, the first five segments are to be replaced with references and the references happen to be R^1_{15}, R^1_{16}, R^1_{17}, R^1_3 and R^1_8. Note that the references are not necessarily in order, and this example illustrates that some references (e.g., R^1_3 and R^1_8.) might be to segment data that were already encountered, in which case a new segment is not used, but the reference is to the preexisting segment.

Ideally, the encoder would then determine, for example, that the sequences (R^1_{15}, R^1_{16}, R^1_{17}) and (R^1_3, R^1_8) recur frequently, so that they can be grouped and replaced with group labels, such as R^2_1, and R^2_2, respectively. However, solving this problem in general is difficult (similar in difficulty to solving the same problem directly on the underlying data). Thus, the encoder re-applies the method of content-induced segmentation to the reference label sequence, yielding similar benefits to the original approach but at a higher-level (i.e., relatively low complexity and the ability to find patterns in the input sequence independent of shifts and localized changes to the underlying data). Thus, supposing the segmenter decided that the sequence (R^2_1, R^2_2, S_F) corresponded to a new segment at this higher layer (by reapplying a fingerprint function, window, and indicator), the original input data can then be represented by a reference to the new segment (R^2_1, R^2_2, S_F). Correspondingly, a binding would be entered into the PSS that related the new reference label to the new higher-level segment. Although not present in this example, the final sequence can have reference labels and group labels from any or all levels. As needed, the segment bindings and the reference bindings are stored and/or provided to the decoder.

Fixed-level HCS

With a fixed-level HCS, the PSS is structured as a set of N (some indeterminate integer greater than one) binding tables PSS^1, PSS^2, . . . , PSS^N. Binding table PSS^1 provides bindings between reference labels and segment data. Binding table PSS^2 provides bindings between reference label sequences and group labels. Other binding tables provide bindings for groups of groups of reference labels, and so on. This is illustrated in FIG. **6**.

The binding tables can store an arbitrary string of bits for each segment and an arbitrary sequence of reference labels. Using the example segmentation and representation from FIG. **5**, PSS^1 would hold the bindings (R^1_{15}, S_A), (R^1_{16}, S_B), (R^1_{17}, S_C), (R^1_3, S_D) and (R^1_{18}, S_E) and PSS^2 would hold the reference bindings (R^2_1, R^1_{15}+R^1_{16}+R^1_{17}) and (R^2_2, R^1_3+R^1_8).

Using this scheme, all the data in the input buffer might be ultimately represented with a single label, R^N_1, and if the data sequence appears again, it will be efficiently represented in a single reference symbol. Likewise, if subsets of the data or slightly altered portions of the data are presented again, the similar portions will be efficiently represented by the same compact symbol string and only the differences will be represented by differing compact symbol strings.

This hierarchical decomposition of the encoding process combined with content-based segmentation has the attractive property that local variations to data, even if arbitrarily large, do not impact the unchanged portions of the repeated data. In a system that used fixed-size segments, every segment would change, for instance, if a few bytes near the front of a data object were inserted or deleted because the data in each fixed-size block would shift and thus look different to the system. In the example of FIG. **5**, however, arbitrary amounts of data could be inserted in the object at the points covered by segment S_C and only references R^1_{17} and R^2_1 would be impacted. For large data objects, this localization of the impact of allows important performance savings to be exploited by various algorithms built on top of this framework.

Variable-level HCS

Instead of the fixed N levels of the prior example, the reference grouping can have a variable depth determined at run time. This eliminates the problem of having to choose a value for N that would work over all types and sizes of data.

As outlined in FIG. **7**, if the number of encoding levels is too small, then the encoded reference stream for a large block of data will still require many symbols, since many reference labels will still be needed to encode in the underlying data. In other words, if there were more levels of hierarchy, then the number of symbols representing the underlying data at the topmost level would be further reduced. However, if the number of encoding levels is too large, as shown in FIG. **8**, then the encoding will introduce unnecessary overhead for small chunks of data, since reference labels will be defined and sent unnecessarily.

FIG. **9** is an illustration of a persistent segment store organized to hold an arbitrary depth of references to address this issue. There, instead of each reference label having a specified level (the superscript in the above example), all the references are treated equally, whether they are reference labels for a sequence of segment data symbols, a sequence of reference labels, or a combination of the two. As shown in FIG. **9**, reference label R_1 is bound to segment data S_A, reference label R_2 is bound to segment data S_B, reference label R_t is bound to a group of reference labels (R_3, R_7, R_9), and so on.

Given that the encoder can flexibly choose appropriate levels of encoding based on the input stream, the decoder should be informed of the number of levels of hierarchy present at any given time. Thus, this information should be somehow conveyed in the coded bit stream. In one embodiment, the coded reference stream explicitly indicates the number of levels, verbatim, in the bit stream with special codes. In this scheme, when the encoder changes the number of levels in use, it would emit a code to explicitly indicate the change.

In an alternative embodiment, the adaptive level structure can be conveyed in the encoded output by marking each segment (as conveyed in a binding) explicitly as being either nonterminal or terminal, as shown in FIG. **9**. There, terminal segments represent strings of the final output data while nonterminal segments represent strings of labels. This information would be stored as a "leaf bit" in the PSS, indicating for each binding, whether it terminates the hierarchy and represents the final output data or whether it refers to a sequence of labels that refer to other bindings. A set leaf bit (e.g., a "1") indicates that the record is for a terminal segment and the contents of the record are segment data without any further delimiters and a cleared leaf bit (e.g., a "0") indicates that the record is a sequence of reference labels. Where reference labels are fixed length or have unique decipherability, no space is needed to delimit the references in the group. In another embodiment, the content of each record is encoded such that a reader can determine whether the content includes references or is all just segment data without the presence of a specific leaf bit.

To implement Variable-level HCS, as the encoder consumes the input data stream, it generates new references and appends them to a sequence of first-level reference labels. Each time a new reference is appended to this growing block of labels, the encoder determines whether a segment bound-

ary should be defined using content-induced segmentation applied to the reference label sequence. Thereby, a fingerprint is computed over the label sequence using a fingerprint window and a fingerprint indicator function dictating where to create a new segment boundary. Note that these parameters are independent of the decoding process and do not need to be known by the decoder (rather, they control performance tradeoffs at the encoder). If the fingerprint function indicates that the newly appended reference does not cause the sequence to define a segment boundary, the encoder continues consuming the next input data symbol.

However, if the encoder does detect a new segment boundary in the first-level reference sequence, a new segment is defined comprising the string of reference labels. Similarly to the process that occurs when a new segment is defined in the input data, this new segment is compared against all the existing segments in the appropriate PSS. If an existing segment is found, the existing reference label for that segment is retrieved from the PSS. If no segment is found, a new label is assigned to the block of reference labels and a new binding is added to the PSS (and potentially the output stream). In either case, the second-level label can now be used to express the sequence of first-level reference labels. This new label is then appended to a growing sequence of second-level reference labels. The encoding process examines this second-level reference sequence, applying content-induced segmentation to the second-level sequence of reference labels, again determining if there is a segment boundary and if so, generating a third-level reference for the segment of second-level labels. This process repeats incrementally for subsequent levels of hierarchy, in each case "bubbling up" new reference definitions to the next higher level. In this manner, a large block of data passing through the encoder will likely pass through the recursion multiple times, whereas a small block of data would not undergo unnecessary encoding levels.

FIG. **10** illustrates a decoding process using the persistent segment store of FIG. **9**, wherein each reference label is replaced with the content of its binding. If the segment content contains a sequence of references, these references are in turn replaced, and the process repeats until all of the reference labels refer to terminal nodes and then the segment data can be output. In other words, the decoder recursively resolves reference blocks, terminating when it reaches a data segment. In this embodiment, the encoded stream is equivalent to a hierarchical tree of references where the segments at the leaves of the tree are marked (via the terminal flag) to indicate where the decoding traversal should halt.

One of the benefits of this segmentation scheme is that common patterns within a single input stream can be effectively recognized and compressed. FIG. **11** illustrates one such example. The input stream contains an initial sequence of data bytes that is cut into three segments: S_1, S_2, and S_3. These segments are allocated references R_1, R_2, and R_3. In this example, the remaining input stream contains, among other symbols, the same data repeated twice more. Because the segment cut points are defined as a function of the content, the same segment boundaries will be detected and found in the PSS, and so the same sequence of references will be output. Similarly, assume the hierarchical encoding determines that the sequence of references $<R_1, R_2, R_3>$ defines a cut point for a new segment, labeled R_{10}, then again, the single label R_{10} can be used to identify the repeated sequence later on in the input stream.

HCS-enabled Client-Server Transport Proxy

FIG. **12** illustrates an example of a system where HCS coding described herein might be used. As shown there, a client **612** is coupled to a server **614** over a network **616**, via a client-side transaction accelerator ("CTA") **620** and a server-side transaction accelerator ("STA") **622**. While only one client and one server are shown, it should be understood that the CTA and STA might well operate with multiple clients and/or multiple servers.

Client **612** is coupled to a client proxy **630** of CTA **620**. The other elements of CTA **620** shown in FIG. **12** include a transaction transformer (TT) **632**, an inverse transaction transformer (TT^{-1}) **634**, a persistent segment store (PSS) **636** and a reference resolver (RR) **638**. Server **614** is coupled to a server proxy **640** of STA **622**, which is shown including elements similar to those of CTA **620**, such as a transaction transformer (TT) **642**, an inverse transaction transformer (TT^{-1}) **644**, a persistent segment store (PSS) **646** and a reference resolver (RR) **648**.

Client **612** is coupled to client proxy **630**, which is coupled to TT **632** and TT^{-1} **634**. TT **632** is coupled to PSS **636** and to the network between CTA **620** and STA **622**. TT^{-1} **634** is coupled to PSS **636**, client proxy **630**, RR **638** and to the network between CTA **620** and STA **622**. RR **638**, as shown, is also coupled to PSS **636** and to the network between CTA **620** and STA **622**.

On the other side of the figure, server **614** is coupled to server proxy **640**, which is coupled to TT **642** and TT^{-1} **644**. TT **642** is coupled to PSS **646** and to the network between STA **622** and CTA **620**. TT^{-1} **644** is coupled to PSS **646**, server proxy **640**, RR **648** and to the network between STA **622** and CTA **620**. RR **648**, as shown, is also coupled to PSS **646** and to the network between STA **622** and CTA **620**.

Of the connections shown, arrows indicate the most common direction or directions of flow of information, but information could flow in additional directions and information flow in a single direction might involve data flowing in the reverse direction as well. For example, TT **632** generally sends information in the direction of TT^{-1} **644**, but that might involve data such as confirmations, handshakes, etc., flowing from TT^{-1} **644** to TT **632**.

In operation, the CTAs and STAs segment the payloads of their transactions where warranted and store/cache strings or other sequences of data ("segments") derived from those payloads using a unique naming scheme that can be independent of the transaction and when sending the payload from one TA to another, substitute references to the segments for the segment data when the segment data is such that the sender can expect that the receiver would have that uniquely named segment data, either because it appeared in an earlier transaction or was sent through other processes to the receiver.

For example, the client-server interactions may involve the transfer of file data from the server to the client, though this is just one special case of a client-server transaction-oriented application. Here, client **612** might request a number of files from server **614** by sending file open requests to server **614**. These requests would be proxied by client proxy **630**, which would interact with STA **622**, which in turn proxies the requests to server **614** via its proxy **640**. In this manner, the client-server communications can be accelerated in a manner transparent (except for performance improvements) to the other entity. Client proxy **630** routes the requests to TT **632**, which being file open requests, are probably not encoded. The response from server **614** is assumed to contain a payload comprising a portion of the request files. In some systems, a server first responds to a file

request message with a short response and includes a payload only when the client makes a file read request on an open file. Here, we assume that the server is returning file contents as payload.

When server proxy **640** receives the response message and its payload, it conveys the data to TT **642**, which encodes the response message (or just its payload), as is described below, and adds any bindings it creates to PSS **646**, if they do not already exist, and sends the encoded message to TT^{-1} **634**, which reconstructs the message and relays it to client **612** via client proxy **630**. TT^{-1} **634** can reconstruct the message because it has the bindings to replace segment references in the message. Those bindings come from PSS **636** and where they are not in PSS **636**, reference resolver **638** can get the bindings from reference resolver **648**. In this manner, messages can be encoded and decoded transparent to the message sender and recipient.

As each TT creates bindings, it assigns globally unique reference labels so that a recipient of a reference label will always be able to replace it with unambiguous segment data. Several schemes are usable to ensure unique names, as described in McCanne I.

HCS-enabled File System

Another embodiment of the present invention uses HCS within a file system. By properly applying HCS to the design of a file system, substantial gains in capacity can be achieved by recognizing patterns of repeated data across all the files (and data structures) in a file system and, accordingly, storing any given pattern of data just once on disk. One approach to leveraging HCS in a file system in this fashion is to incorporate the HCS Encoder and HCS Decoder as native components of a native file system. For example, whenever the file system reads or writes a disk block, HCS could be applied to these disk blocks. However, this direct approach may introduce performance problems as many applications require high-performance file I/O but the HCS coding processes could introduce computational overhead. Also, HCS works most efficiently when presented with large, contiguous regions of application or file data and many implementations of file systems typically manipulate files using relatively small, fixed-size blocks.

A better approach is to apply HCS to a file system that employs whole-file accesses rather than block-based accesses. The system shown in FIG. **13** implements a file system with a whole-file interface and leverages HCS. That is, clients store, retrieve, and delete entire files instead of individual pieces of files. While this approach typically could not support a standard file-system interface, it is well suited for other uses of file systems, e.g., as the foundation of a backup system. A disk-based, on-line backup system, where access is slower than a native file system but faster than a traditional backup system (e.g., based on tape), is often called "near-line" storage to describe the compromise between a high-performance, on-line file system and low-performance, off-line backup system. Thus, herein, the file system depicted in FIG. **13** is called a "near-line file system (NLFS)".

In the model shown, a client can store, retrieve, and delete whole files. The NLFS employs an HCS Encoder, PSS, and HCS Decoder to efficiently store file data such that only a single copy of each common segment need be stored once. In addition, the NLFS employs a data base table, called the "file map data base (FMDB)", to map file names to the top-level label (or labels) that represents a particular file. In one embodiment, the FMDB includes a time stamp field so that the file system may contain multiple copies of the same file that represent the file at different points in time.

The system functions as follows. A client stores or updates a file by sending a "STORE" command to the file system followed by the file name and file data. The file system then transmits all of the file data to the HCS Encoder, which returns one or more labels that represent the stored file. Those labels are then entered into the FMDB as part of the tuple representing the new or updated file. Included in this tuple is the corresponding file name and current time stamp (taken from a time-of-day clock, for example).

To retrieve a file, a client sends a "RETRIEVE" command to the file system along with the filename and optional time stamp. The file system, in turn, sends a query to the FMDB with the file name as the key, retrieving all tuples that contain said file name. If there are no such tuples, an error is returned to the client. The file system then selects the tuple with the greatest time stamp that is less than or equal to the specified time stamp. If no time stamp is specified in the command, the tuple with the greatest time stamp is chosen. This has the effect of choosing the version of the file that was stored in the system at the requested time. Next, the file system sends the label string from the selected tuple to the HCS Decoder. The HCS Decoder, in turn, decodes the labels into file data and returns that data to the file system. The file system then returns the file data to the client. Of course, this process could be pipelined such that the file system could deliver file data to the client as the decoder delivers the decoded file to the file system.

To delete a file from the file system, a client sends a "DELETE" command to the file system along with the filename and an optional time range, instructing to delete all files that were stored with respect to the time range specified. In response, the NLFS looks up and removes the corresponding entries from the FMDB and deletes the corresponding labels from the PSS.

In another embodiment, HCS can be used as a complementary component of a more standard, operating system-based file system by combining the NLFS described above with a file system front-end. This approach is depicted in FIG. **14**, which shows NLFS and a file server front-end. In this approach, whenever one or more clients open a particular file, the file server reads the file from the NLFS system and copies the file into a local file system. This file is effectively cached in the local front-end file system, which allows the file server to employ normal file system protocols and algorithms for managing the open file. By ensuring that only the front-end file server communicates with the NLFS and by employing normal file system locking and consistency mechanisms, there is no possibility that the files stored in the NLFS become inconsistent with those in the file server's local file system. When all clients close the file, the file server writes back the file to the NLFS and frees up the corresponding storage in the local file server.

In another embodiment, HCS can be used as a complementary component of a more standard, operating system-based file system by combing the NLFS described above with a file system front-end. This approach is depicted in FIG. **14**, which shows NLFS and a file server front-end. In this approach, whenever one or more clients open a particular file, the file server reads the file from the NLFS system and copies the file into a local file system. This file is effectively cached in the local front-end file system, which allows the file server to employ normal file system protocols and algorithms for managing the open file. By ensuring that only the front-end file server communicates with the NLFS and by employing normal file system locking and consis-

tency mechanisms, there is no possibility that the files stored in the NLFS become inconsistent with those in the file server's local file system. When all clients close the file, the file server writes back the file to the NLFS and frees up the corresponding storage in the local file server.

There are many ways to build the front-end file server based on well-known techniques for file system design and whole file caching, but in one embodiment, the well-known unix file system (UFS) is extended to interface with NLFS. For example, UFS (as well as other file systems) uses directories to map file names to disk data structures. In UFS, the directory entry for a file points to an index node (or inode). UFS, thus, could be extended with a new inode type that contains a reference to a file in NLFS. Under this scheme, when a client opens an existing file, UFS checks for the new inode type and when this is the case, reads the file from the NLFS. UFS then writes the returned file data in the local file system effectively formatting the file data onto a local disk using normal inode data structures. In effect, the single inode pointing to an NLFS file is replaced with a multitude of inodes and disk blocks to represent the actual file locally.

At this point, the client open call finishes. Thereafter, all client file activity proceeds identically to the traditional UFS model (including cases where multiple clients open the same file simultaneously) and any file updates or changes are reflected in the local file copy. When the file is closed (by all clients), the modified UFS then reads the file data from local file system (using the normal inode data structures) and writes this data back to the NLFS. When the file is successfully written, the inodes representing the file data can be freed up and replaced with a single inode of the new type that points to NLFS. Alternatively, the file can be copied out the NLFS in this fashion without necessarily freeing up the local copy of the file so that future accesses to the file do not need to wait for NLFS. Then, later, when the local file system begins to fill to capacity, the UFS could free up files using the method describe above by applying any number of the numerous well-known cache replacement policies (e.g., least recently used, least frequently accessed, and so forth).

The above description is illustrative and not restrictive. Many variations of the invention will become apparent to those of skill in the art upon review of this disclosure. The scope of the invention should, therefore, be determined not with reference to the above description, but instead should be determined with reference to the appended claims along with their full scope of equivalents.

What is claimed is:

1. A method for storing data, comprising:
receiving a file from a client at a server;
segmenting the file into one or more segments;
forming a list, comprising:
determining whether each segment is present in a segment store, wherein a segment that is present in the segment store has an assigned reference label;
for each segment present in the segment store, adding to the list the assigned reference label; and
for each segment not present in the segment store, assigning a reference label to the segment, storing the segment and the reference label, and adding the reference label to the list; and
storing an association between the file and the list.

2. The method of claim **1**, wherein the segment and its assigned reference label are both stored in the segment store.

3. The method of claim **1**, wherein the reference labels are partly independent of content of a segment.

4. The method of claim **1**, further comprising adding a time stamp to the list.

5. The method of claim **1**, wherein the server is a unix file system (UFS), and wherein a directory entry in the UFS for a file references the segment store.

6. The method of claim **1**, further comprising:
determining whether any sequence of segments is grouped as a reference group,
wherein forming a list further comprises:
determining whether a reference group is present in the segment store, wherein the segment store includes a group label to the reference labels of the reference group present in the segment store;
for each reference group present in the segment store, adding to the list the group label; and
for each reference group not present in the segment store, assigning a reference label to a reference group segment not present in the segment store, storing the reference group segment and the reference label in the segment store, assigning a group label to the reference labels of the reference group, storing the group label in the segment store, and adding the group label to the list.

7. A system for storing data, comprising:
a server for receiving a file from a client;
a segmenter configured to segment the file;
an encoder for forming a list, comprising:
logic configured to determine whether each segment is present in a segment store, wherein a segment that is present in the segment store has an assigned reference label;
for each segment present in the segment store, logic configured to add to the list the assigned reference label; and
for each segment not present in the segment store, logic configured to assign a reference label to the segment, store the segment and the reference label, and add the reference label to the list; and
logic configured to store an association between the file and the list.

8. The system of claim **7**, wherein the segment and its assigned reference label are both stored in the segment store.

9. The system of claim **7**, wherein the reference labels are partly independent of content of a segment.

10. The system of claim **7**, further comprising logic configured to add a time stamp to the list.

11. The system of claim **7**, wherein the server is a unix file system (UFS), wherein a directory entry in the UFS for a file references the segment store.

12. The system of claim **7**, further comprising:
a sequence determiner for determining whether any sequence of segments is grouped as a reference group,
wherein the encoder further comprises:
logic configured to determine whether a reference group is present in the segment store, wherein the segment store includes a group label to the reference labels of the reference group present in the segment store;
for each reference group present in the segment store, logic configured to add to the list the group label; and
for each reference group not present in the segment store, logic configured to assign a reference label to a reference group segment not present in the segment store, store the reference group segment and the reference label in the segment store, assign a group label to the reference labels of the reference group,

store the group label in the segment store, and add the group label to the list.

13. A method for retrieving data, comprising:

receiving a request for a file from a client at a server;

retrieving a list including reference labels to segments of the file, wherein the list is associated with the file;

decoding the list, comprising:

replacing reference labels with segments of the file; and

sending the file from the server to the client.

14. The method of claim **13**, wherein the reference labels are partly independent of content of a segment.

15. The method of claim **13**, wherein the retrieved list has a time stamp that is greatest time that other lists associated with the file.

16. The method of claim **13**, wherein the request includes a specified time stamp, and wherein the retrieved list has the greatest time stamp that is less than or equal to the specified time stamp.

17. The method of claim **13**, wherein the server is a unix file system (UFS), wherein a directory entry in the UFS for a file references the segment store.

18. The method of claim **13**, wherein the list includes a group label to reference labels to segments of the file, and wherein decoding the list further comprises replacing a group label with the segments to which the reference labels refer.

19. A system for storing files, comprising:

a front-end file system for receiving from a client a file command for a file, wherein the front-end file system includes a front-end file server;

a back-end storage system including

logic to segment the file; and

a segment store for storing the segments;

a network file system interface for sending or receiving contents of the file between the front-end file system and the back-end storage system.

20. The system of claim **19**, wherein the network file system interface is NFS.

21. The system of claim **19**, wherein the network file system interface is CIFS.

22. The system of claim **19**, wherein the front-end file system caches file data from the segment store when the file is being accessed by the client.

23. The system of claim **19**, wherein the file is constructed from data received at the front-end file system from the segment store when the client opens the file.

24. The system of claim **19**, wherein the file is sent from the front-end file system to a representation in the segment store when the client closes the file.

25. A method for storing files, comprising:

receiving, at a front-end file system from a client, a file command for a file, wherein the front-end file system includes a front-end file server;

based on the file command, using a network file system interface to transfer contents of the file between the front-end file system and a back-end storage system;

segmenting the file at the back-end storage system; and

maintaining a storage of the segments in a segment store of the back-end storage system.

26. The system of claim **25**, wherein the network file system interface is NFS.

27. The system of claim **25**, wherein the network file system interface is CIFS.

28. The method of claim **25**, further comprising caching file data from the segment store at the front-end file system when the file is being accessed by the client.

29. The method of claim **25**, further comprising constructing the file from data received at the front-end file system from the segment store when the client opens the file.

30. The method of claim **25**, further comprising sending the file from the front-end file system to a representation in the segment store when the client closes the file.

* * * * *

EXHIBIT C

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

QUANTUM CORPORATION,

Plaintiff,

v.

RIVERBED TECHNOLOGY, INC.,

Defendant.
_____/

AND RELATED COUNTERCLAIMS.
_____/

No. C 07-04161 WHA

CASE MANAGEMENT ORDER AND REFERENCE TO MAGISTRATE JUDGE FOR MEDIATION/SETTLEMENT

After a case management conference, the Court enters the following order pursuant to Rule 16 of the Federal Rules of Civil Procedure ("FRCP") and Civil Local Rule 16-10:

1. All initial disclosures under FRCP 26 must be completed by **DECEMBER 7, 2007**, on pain of preclusion.

2. Leave to add any new parties or pleading amendments must be sought by **DECEMBER 7, 2007**.

3. The claim-construction hearing will be held on **MAY 7, 2008**, at **1:30 P.M.** Counsel shall meet and confer and propose a briefing schedule leading up to the hearing. This must be filed by **DECEMBER 28, 2007**, and entitled "PROPOSED ORDER RE SCHEDULE FOR CLAIM CONSTRUCTION" and send it by email to whapo@cand.uscourts.gov.

4. For the claim-construction hearing, the parties must isolate no more than **SIX PHRASES** in all claims at issue for all patents and limit the hearing to those phrases. All other phrases in dispute will be determined at trial absent further order. In the Court's experience, most patent cases turn on the meaning of only a few phrases. Once defined, the case will usually resolve by motion or settlement. If it does not, then the Court will construe any remaining phrases at issue during summary judgment or, at a minimum, before the jury is instructed. Counsel must conduct the presentations at the claim-construction hearing. Expert testimony will normally not be needed but all sides may have an expert present to address points outside the intrinsic record should they arise. In the briefing on claim construction, please include a precis of the eventual summary-judgment issues and how claim construction differences may affect summary judgment. A tutorial for the Court (to be conducted by counsel only, not experts) shall be set for **APRIL 23, 2008**, at **1:30 P.M.**

5. The non-expert discovery cut-off date shall be **SEPTEMBER 26, 2008**.

6. The deadline for producing opinions of counsel under Patent Local Rule 3-8 shall be **28 CALENDAR DAYS** before the non-expert discovery cut-off, irrespective of the timeline in said rule.

7. The last date for designation of expert testimony and disclosure of full expert reports under FRCP 26(a)(2) as to any issue on which a party has the burden of proof ("opening reports") shall be **SEPTEMBER 26, 2008**. Within **FOURTEEN CALENDAR DAYS** thereafter, all other parties may disclose responsive expert testimony with full expert reports responsive to opening reports ("opposition reports"). Within **SEVEN CALENDAR DAYS** thereafter, the opening parties may disclose any reply reports limited solely to rebutting specific material in opposition reports. Reply reports must be limited to true rebuttal and should be very brief. They should not add new material that should have been placed in the opening report. The cutoff for all expert discovery shall be **FOURTEEN CALENDAR DAYS** after the deadline for reply reports. In aid of preparing an opposition or reply report, a responding party may depose the adverse

expert sufficiently before the deadline for the opposition or reply report so as to use the testimony in preparing the response. Experts must make themselves readily available for such depositions. Alternatively, the responding party can elect to depose the expert later in the expert-discovery period. An expert, however, may be deposed only once unless the expert is used for different opening and/or opposition reports, in which case the expert may be deposed independently on the subject matter of each report. At least **28 CALENDAR DAYS** before the due date for opening reports, each party shall serve a list of issues on which it will offer any expert testimony in its case-in-chief (including from non-retained experts). This is so that all parties will be timely able to obtain counter-experts on the listed issues and to facilitate the timely completeness of all expert reports. Failure to so disclose may result in preclusion.

8. As to damages studies, the cut-off date for *past damages* will be as of the expert report (or such earlier date as the expert may select). In addition, the experts may try to project *future damages* (*i.e.*, after the cut-off date) if the substantive standards for future damages can be met. With timely leave of Court or by written stipulation, the experts may update their reports (with supplemental reports) to a date closer to the time of trial.

9. At trial, the direct testimony of experts will be limited to the matters disclosed in their reports. Omitted material may not ordinarily be added on direct examination. This means the reports must be complete and sufficiently detailed. Illustrative animations, diagrams, charts and models may be used on direct examination only if they were part of the expert's report, with the exception of simple drawings and tabulations that plainly illustrate what is already in the report, which can be drawn by the witness at trial or otherwise shown to the jury. If cross-examination fairly opens the door, however, an expert may go beyond the written report on cross-examination and/or redirect examination. By written stipulation, of course, all sides may relax these requirements.

10. To head off a recurring problem, experts lacking percipient knowledge should avoid vouching for the credibility of witnesses, *i.e.*, whose version of the facts in dispute is correct. This means that they may not, for example, testify that based upon a review of fact depositions and other material supplied by counsel, a police officer did (or did not) violate standards. Rather, the expert should be asked for his or her opinion based — explicitly — upon an assumed fact scenario. This will make clear that the witness is not attempting to make credibility and fact findings and thereby to invade the province of the jury. Of course, a qualified expert can testify to relevant customs, usages, practices, recognized standards of conduct, and other specialized matters beyond the ken of a lay jury. This subject is addressed further in the trial guidelines referenced in paragraph 17 below.

11. The last date to file dispositive motions shall be **NOVEMBER 6, 2008**. No dispositive motions shall be heard more than 35 days *after* this deadline, *i.e.*, if any party waits until the last day to file, then the parties must adhere to the 35-day track in order to avoid pressure on the trial date.

12. The **FINAL PRETRIAL CONFERENCE** shall be at **2:00 P.M.** on **JANUARY 12, 2009**. For the form of submissions for the final pretrial conference and trial, please see paragraph 17 below.

13. A **JURY TRIAL** shall begin on **FEBRUARY 2, 2009**, at **7:30 A.M.**, in Courtroom 9, 19th Floor, 450 Golden Gate Avenue, San Francisco, California, 94102. The trial schedule and time limits shall be set at the final pretrial conference. Although almost all trials proceed on the date scheduled, it may be necessary on occasion for a case to trail, meaning the trial may commence a few days or even a few weeks after the date stated above, due to calendar congestion and the need to give priority to criminal trials. Counsel and the parties should plan accordingly, including advising witnesses.

14. Counsel may not stipulate around the foregoing dates without Court approval.

15. While the Court encourages the parties to engage in settlement discussions, please do not ask for any extensions on the ground of settlement discussions or on the ground that

the parties experienced delays in scheduling settlement conferences, mediation or ENE. The parties should proceed to prepare their cases for trial. No continuance (even if stipulated) shall be granted on the ground of incomplete preparation without competent and detailed declarations setting forth good cause.

16. To avoid any misunderstanding with respect to the final pretrial conference and trial, the Court wishes to emphasize that all filings and appearances must be made — on pain of dismissal, default or other sanction — unless and until a dismissal fully resolving the case is received. It will not be enough to inform the clerk that a settlement in principle has been reached or to lodge a partially executed settlement agreement or to lodge a fully executed agreement (or dismissal) that resolves less than the entire case. Where, however, a fully-executed settlement agreement clearly and fully disposing of the entire case is lodged reasonably in advance of the pretrial conference or trial and only a ministerial act remains, the Court will arrange a telephone conference to work out an alternate procedure pending a formal dismissal.

17. If you have not already done so, please read and follow the "Supplemental Order to Order Setting Initial Case Management Conference in Civil Cases Before Judge William Alsup" and other orders issued by the Clerk's office when this action was commenced. Among other things, the supplemental order explains when submissions are to go to the Clerk's Office (the general rule) versus when submissions may go directly to chambers (rarely). With respect to the final pretrial conference and trial, please read and follow the "Guidelines For Trial and Final Pretrial Conference in Civil Jury Cases Before The Honorable William Alsup." All orders and guidelines referenced in the paragraph are available on the district court's website at http://www.cand.uscourts.gov. The website also includes other guidelines for attorney's fees motions and the necessary form of attorney time records for cases before Judge Alsup. If you do not have access to the Internet, you may contact Deputy Clerk Dawn K. Toland at (415) 522-2020 to learn how to pick up a hard copy.

18. This matter is hereby **REFERRED** to **MAGISTRATE JUDGE ELIZABETH D. LAPORTE** for **SETTLEMENT/MEDIATION**, the Court believing that such a conference would be more effective in settling the present case than any other avenue.

19. All pretrial disclosures under FRCP 26(a)(3) and objections required by FRCP 26(a)(3) must be made on the schedule established by said rule.

IT IS SO ORDERED.

Dated: November 29, 2007.

WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

EXHIBIT D

[COUNSEL LISTED ON SIGNATURE PAGE]

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

QUANTUM CORPORATION, Plaintiff, v. RIVERBED TECHNOLOGY INC., Defendant. AND RELATED COUNTERCLAIMS	Case No. C07-04161 WHA **STIPULATION REGARDING DISCOVERY AND ~~[PROPOSED]~~ ORDER** Honorable William H. Alsup United States District Judge

Riverbed Technology, Inc. ("Riverbed") and Quantum Corporation ("Quantum") respectfully request that the Court enter the following stipulation the parties have reached regarding discovery in this action.[1] The parties will adhere to the Court's Supplemental Order Setting Initial Case Management Conference in Civil Cases Before Judge Alsup ("Supplemental Order") (Docket No. 14) and the discovery provisions therein. However, pursuant to Paragraph 12 of the Supplemental Order, the parties hereby agree to clarify and modify certain aspects of the Supplemental Order as follows:

1. **Privilege Log Requirements.**

The parties are not required to list the following documents or communications on their privilege logs: (1) documents or communications dated after Quantum's initial filing of this lawsuit; (2) intra-firm documents or communications (*e.g.*, communications taking place within litigation counsel's law firm) generated before or after Quantum's initial filing of this lawsuit; and (3) communications between a party and its opinion counsel that occurred after the obtainment of any final opinion letters regarding patent issues relevant to this action.

Each party shall provide a privilege log within two weeks of the date of its first initial obligation to respond to document requests in this action. The initial privilege log shall list all documents or communications for which a claim of privilege or work product is made that the party is aware of at the time the initial privilege log is produced. The parties shall promptly supplement their initial privilege log if they become aware of additional documents or communications subject to a claim of privilege or work product. Privilege logs shall provide the information required sections (a)-(d) in Paragraph 17 of the Supplemental Order. Adherence to this procedure shall not be deemed a waiver of any privilege or protection.

---

[1] Several of the following discovery agreements were included in the parties' Joint Case Management Statement and raised during the November 29, 2007 Case Management Conference. During that conference, the Court indicated that such agreements between the parties were permissible and would be enforced by the Court if in writing.

Paragraph 17(b) of the Supplemental Order may be fulfilled by the following statement (or something substantially similar) from the party responding to the discovery requests: "[Party] certifies that adequate steps were taken to ensure the confidentiality of the document or communication and that to the best of [Party's] knowledge no unauthorized persons received the document or communication."

2. **Drafts of Expert Reports.**

Paragraph 23 of the Supplemental Order shall be modified to provide that nothing shall preclude an expert from working on and updating a continuous version of an expert report without preserving interim drafts thereof even if the expert has discussions with litigation counsel while doing so.

3. **Rule 30(b)(6) Depositions.**

Paragraph 24(b) of the Supplemental Order shall be modified to provide that when one individual is deposed in both his capacity as a corporate designee and as an individual, the testimony need not be separately transcribed so long as a clear statement is made on the record, and acknowledged on the record by counsel representing the witness, indicating that the questioning will thereafter be directed to the witness in his individual instead of corporate capacity, or vice versa.

Paragraph 24(b) shall be modified to provide that any deposition pursuant to Federal Rule of Civil Procedure 30(b)(6) shall count as only one deposition, regardless of how many individuals the company-deponent designates to give testimony as to the various topics noticed, unless the testimony of any one designee lasts more than four hours, in which case that designee's testimony will count as a separate deposition against the deposing party's total allotment.

NOW, THEREFORE, the parties to this action, through their respective counsel of record, **AGREE TO AND HEREBY STIPULATE** to the foregoing.

Dated: December 6, 2007

QUINN EMANUEL URQUHART OLIVER & HEDGES, LLP

/s/ Todd M. Briggs

Todd M. Briggs
Attorneys for Defendant and Counterclaimant,
RIVERBED TECHNOLOGY, INC.

Dated: December 6, 2007

SHEPPARD MULLIN

/s/ Amar Thakur

Amar Thakur
Attorneys for Defendant and Counterclaimant
QUANTUM CORPORATION

I, Todd M. Briggs, am the ECF User whose identification and password are being used to file this document. Pursuant to General Order 45.X.B, I hereby attest that counsel for Quantum has concurred in this filing.

ORDER

PURSUANT TO STIPULATION, IT IS SO ORDERED

UNITED STATES DISTRICT COURT NORTHERN DISTRICT OF CALIFORNIA
IT IS SO ORDERED
Judge William Alsup

Dated: December 7, 2007

Honorable William H. Alsup
United States District Judge

[COUNSEL LISTED ON SIGNATURE PAGES]

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

QUANTUM CORPORATION,

Plaintiff,

vs.

RIVERBED TECHNOLOGY, INC.,

Defendant.

RIVERBED TECHNOLOGY, INC.,

Counterclaimant,

vs.

QUANTUM CORPORATION,

Counterdefendant.

CASE NO. C 07-4161 WHA

~~[PROPOSED]~~ ORDER RE SCHEDULE FOR CLAIM CONSTRUCTION

Pursuant to Paragraph 3 of the Court's Case Management Order (Docket No. 23), Riverbed Technology, Inc. ("Riverbed") and Quantum Corporation ("Quantum") jointly submit this Proposed Order Re Schedule For Claim Construction and will email a copy to whapo@cand.uscourts.gov. The parties' proposal includes agreed dates for claim construction briefs and events required by the Patent Local Rules.

EVENT	AGREED DATE
Disclosure of Asserted Claims and Preliminary Infringement Contentions	December 13, 2007
Preliminary Invalidity Contentions	January 31, 2008
Simultaneous Exchange of Proposed Terms and Claim Elements for Construction	February 6, 2008
Simultaneous Exchange of Preliminary Claim Constructions and Extrinsic Evidence	February 20, 2008
Joint Claim Construction and Prehearing Statement	March 12, 2008
Close of discovery related to Claim Construction	March 19, 2008
Opening Claim Construction Briefs	March 26, 2008
Opposition Claim Construction Briefs	April 9, 2008
Reply Claim Construction Briefs	April 16, 2008
Claim Construction Tutorial	April 23, 2008 at 1:30 p.m.*
Claim Construction Hearing	May 7, 2008 at 1:30 p.m.*
Final Infringement Contentions	30 days after issuance of claim construction ruling
Final Invalidity Contentions	50 days after issuance of claim construction ruling

* These dates were set by the Court in its Case Management Order.

NOW, THEREFORE, the parties to this action, through their respective counsel of record, **AGREE TO AND HEREBY STIPULATE** to the foregoing schedule.

Dated: December 6, 2007

QUINN EMANUEL URQUHART OLIVER & HEDGES, LLP

/s/ Todd M. Briggs
Todd M. Briggs
Attorneys for Defendant and Counterclaimant,
RIVERBED TECHNOLOGY, INC.

Dated: December 6, 2007

SHEPPARD MULLIN

/s/ Amar Thakur
Amar Thakur
Attorneys for Defendant and Counterclaimant
QUANTUM CORPORATION

I, Todd M. Briggs, am the ECF User whose identification and password are being used to file this document. Pursuant to General Order 45.X.B, I hereby attest that counsel for Quantum has concurred in this filing.

ORDER

PURSUANT TO STIPULATION, IT IS SO ORDERED

Dated: December 7, 2007

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
IT IS SO ORDERED
Judge William Alsup

Honorable William H. Alsup
United States District Judge

EXHIBIT E

Pages 1 - 25

United States District Court

Northern District of California

Before The Honorable William Alsup

Quantum Corporation, Plaintiff, vs. Riverbed Technology, Incorporated, Defendant.	)	No. C07-4161 WHA

San Francisco, California
Thursday, January 31, 2008

Reporter's Transcript Of Proceedings

Appearances:

For Plaintiff:
Sheppard, Mullin, Richter & Hampton
12275 El Camino Real, Suite 200
San Diego, California 92130
By: Mauricio Flores, Esquire
Amardeep Lal Thakur, Esquire

For Defendant:
Quinn Emanuel Urquhart Oliver & Hedges
555 Twin Dolphin Drive, Suite 560
Redwood Shores, California 94065
By: Claude M. Stern, Esquire
Todd Michael Briggs, Esquire

Reported By:
Sahar McVickar, RPR, CSR No. 12963
Official Reporter, U.S. District Court
For the Northern District of California

(Computerized Transcription By Eclipse)

Thursday, January 31, 2008 8:00 a.m.

P R O C E E D I N G S

THE CLERK: Civil action number C07-4161, Quantum versus Riverbed Technology.

THE COURT: Well, who do we have?

MR. FLORES: Mauricio Flores and Amardeep Thakur on behalf of plaintiff, Quantum.

MR. STERN: Good morning, Your Honor.

Claude Stern and Todd Briggs of Quinn Emanuel on behalf of Riverbed.

Mr. Briggs will be arguing the motion.

THE COURT: Okay.

This is a motion by Riverbed; that's Riverbed over here?

MR. BRIGGS: Yes.

THE COURT: Okay. It's your motion, so go ahead.

MR. BRIGGS: Thank you, Your Honor.

First off, I would just like to say that I appreciate your order from last week requesting that the junior associates be given an opportunity to argue before you.

THE COURT: Well, are you a junior associate?

MR. BRIGGS: I'm a senior associate, but I'm more junior.

THE COURT: Closer to what I wanted than Mr. Stern.

MR. BRIGGS: By many years. By many years, yes.

Thank you.

THE COURT: I'll make my speech again; I worry for the future. After Mr. Stern is long gone and has retired to his 50-acre ranch on the Big Island in Hawaii and all the rest of you are gone, I worry for the future of our federal system, because we are not training a generation to come behind them. And in order to train them, in order for this public to have confidence in our system, it's more important, really, that we have good lawyers than good judges. You need a good judge, but the lawyers are more than two-thirds of the process.

And if the public does not have confidence that the system works, they will say, oh, let's go to JAMS, oh, let's get a mediator, we can't trust a jury. And the real reason is going to be because the lawyers are afraid and unable to do it.

And in order to -- big firms are up against clients that insist on Mr. Big or Ms. Big, and somehow the profession has got to address this issue. I'm not going to solve it by myself, but I'm going to do my little piece of it.

Now, I want to assure you it will make no difference in the outcome of this motion who argues it, that is a separate point. I -- that's not going to be a factor. So I just make that point for all of you to think about as you go forward with your careers and wonder about the future of our system.

All right, enough said on my preaching, let's go to

your motion.

MR. BRIGGS: Okay.

I would like to get right to the question that you presented to us earlier this week, and that is, under the applicable law does a parent company automatically have authority to act as agent to transfer assets of its subsidiaries. And we think under any law that would apply to this case, the answer to that is clearly no.

THE COURT: What law does apply to this issue?

MR. BRIGGS: Well --

THE COURT: Let me give you three possibilities.

MR. BRIGGS: Well, there is --

THE COURT: Federal common law, Australia law, maybe California law, maybe something else.

MR. BRIGGS: Right.

THE COURT: What law does apply to this issue?

MR. BRIGGS: Well, you know, ACN120 and Rocksoft, they are both Australian companies, so if you were looking at an agency theory, if you are determining if one Australian company can act as an agent of another Australian company, it very well may be Australian law. But it could be federal law or it could be California law. We haven't had an opportunity to fully research that issue yet, so I'm not sure, exactly which law.

THE COURT: It is your motion.

MR. BRIGGS: That is correct, but --

THE COURT: So you know, here we are at the time and place to stand and deliver. And I'm not giving anybody more opportunity to do anything.

MR. BRIGGS: Well, the answer is we have done some research after receiving your question under each of those bodies of law, and the answer is no, under each of the -- under each body of law.

THE COURT: Okay, well, lay it on me, as they say, and tell me what the -- tell me each body of law, and then tell me what the -- what you are relying on.

MR. BRIGGS: Sure.

It's a basic tenet of corporate law here in the United States and in Australia that corporations are separate legal entities. They own their own property, and they act -- they act independently of one another. One corporation, just because it owns shares of another corporation, does not mean it has legal title to the assets of that corporation and it can do what it wants with that. And I have a Supreme Court case that states that, Dole versus Patrickson, 538 US 468 from 2003.

And it says here, "A corporate parent which owns the shares of a subsidiary does not for that reason alone own or have legal title to the assets of the subsidiary. And it follows with even greater force that the parent does not own or have legal title to the subsidiaries of the subsidiary."

I have a similar statement under California law.

This is from Northwest Pacific Railroad versus the State Board of Equalization, 21 Cal 2d, 524. The quote from the case: "Ownership of capital stock in one corporation by another does not itself create an identity of corporate interest between the two companies nor render the stock holding company the owner of the property of the other nor create the relation of principle and agent, representative or alter ego between the two."

We also contacted counsel in Australia and corporate counsel in Australia and got a similar answer. Here is a case from -- from Australia, a quote from the case is, "I say that because each of those five matters relate to control, and control of itself cannot be a decisive indicator of agency. If it were otherwise, there would be -- there would often be an agency between a parent company and its subsidiary or a sole shareholder and his company. Such a result is not only inconsistent with Solomon V. Solomon and co-limited, but also with the high court authority, which recognizes this separate legal existence of companies in a group." That is from ACN007528207 Party Limited versus Bird Cameron Partners.

We also found that this is true under Delaware law, if Delaware law were somehow found to apply to this action. So all the authorities are -- are consistent. Any authority that would apply in this case is consistent, and it says that the property of a subsidiary is not -- it's not owned by the

shareholders of the parent corporation.

And so the answer to your question is no. I mean, if the shareholders -- even if it was a sole shareholder who tried to transfer the assets, the board of the subsidiary corporation would have the legal power to block that. So there is no automatic authority to just transfer assets of a subsidiary.

THE COURT: Have you read the motion to supplement the record that was made by the plaintiff? It was filed -- a supplemental declaration?

MR. BRIGGS: Yes.

THE COURT: What's your response to that?

MR. BRIGGS: Well, I don't think that solves any problems here. That just says that -- that is signed by Shawn Hall. Now, Shawn Hall is apparently the C -- not the CEO, but the general counsel of Quantum Corporation, so he wears that hat. Apparently, he is also a director in ACN120 and a director in Rocksoft, along with several other directors. I think Rocksoft has a total of four directors and ACN120 has three directors.

First of all, my first response to it is is that whole declaration is based on an inadmissible hearsay statement. He claims that the CEO of Quantum told him to transfer all substantial rights in the '810 patent before the filing of this lawsuit, to transfer those rights to Quantum. Now, even if that were true, I don't believe he has the power,

acting alone, to do so. There is no evidence that any of these other directors knew of this, any of the other directors at Rocksoft or any of them consented to this. There is no evidence of what the bylaws of Rocksoft are and whether he could unilaterally do this on his own. There is just no evidence. They haven't put forth any of that evidence. And it was their burden on this motion to show they had standing and they didn't meet it.

So you know, based on the record, I don't think you can deduce that he had the authority to do anything on behalf

of Rocksoft and transfer any of their assets.

THE COURT: Let me ask a different question.

You have a counter-claim in this case.

MR. BRIGGS: That's correct.

THE COURT: If the Court grants this motion so the original complaint goes away, do the plaintiffs have the right, in light of your counter-claim, to file a counter-claim to the counter-claim and reassert all of the -- reassert the very same patent now that they have possibly fixed it up with that subsequent assignment?

MR. BRIGGS: Well, first of all, we do have a counter-claim. And if you did dismiss their case, our counter-claim would survive. That was addressed in a Federal Circuit case we cited in our opening brief.

THE COURT: That's the problem. That's why I'm

raising it.

MR. BRIGGS: And I do believe --

THE COURT: I'm not saying what the answer would be, but why are we going through this exercise if, and that is an if, if it's easy enough to fix this just by filing a counter-claim to the counter-claim?

MR. BRIGGS: Well, they could file a counter-claim to the counter-claim, but we have another action pending, we have a declaratory judgment action pending in Delaware. And that would be the first filed action, and so we should be able to proceed with that as the first filed action.

THE COURT: Why don't you just agree that if I grant this motion you will instantly dismiss your counter-claim and reassert it wherever you want to so that all you go away and go to Delaware?

MR. BRIGGS: Well, I mean, we believe we would like to move forward with our claim in this Court, and we believe we can move forward with their claim in Delaware. It provides a strategic advantage for us in the litigation, I mean, to be frank with you.

THE COURT: I appreciate your frankness, but that's why judges are almost fed up with patent lawyers who are always trying to yank the courts around for some strategic advantage

when we got hundreds and hundreds of cases to deal with. You think about that.

10

It would make it a lot easier to grant your motion -- I'm going to grant the motion or deny the motion on the merits, believe me, but I -- I wonder what the point is if -- if, and I don't know what the answer to this counter-claim to a counter-claim point is, but if it's as easy to fix, they can just file a counter-claim because you filed a counter-claim, that's not so easy to say that, well, the first filed case in Delaware will be the one that goes forward.

Okay, thank you, Mr. Briggs.

MR. BRIGGS: You're welcome.

THE COURT: Let's hear from the other side.

MR. FLORES: Good morning, Your Honor.

My name is Mauricio Flores. I'm afraid I'm many years past the lawyer you wanted, but I appreciate your remarks.

THE COURT: You are welcome, anyway. I know that you will take to heart my comments and do the right thing in some other hearing.

All right, go ahead.

MR. FLORES: Indeed, we will.

Your Honor, this case -- this motion isn't decided on the broad proposition of law. We don't assert, to be clear, we do not assert that mere stock ownership gives one the right to --

THE COURT: Then what does? Let's look at that

11

August '07 -- see if I can find it again.

Let me get my law clerk to come up here. Would you come fish out the '07 --

I looked at it; look, here is the problem you have: The guy that signs it signs it under a block that says ACN120, it doesn't say Riverbed.

MR. FLORES: That's correct.

THE COURT: That's the whole problem you got.

MR. FLORES: Yes.

THE COURT: You did this one day before you filed

the lawsuit, and this was done in Australia.

This was more gimmickry by the patent lawyers in this district -- or, actually, you are not even in this district. It was more gimmickry by patent lawyers to set up a lawsuit. And you did it too quick.

So I'm looking at the key page right here: ACN10 signed by Sean somebody. I'm going to assume for the sake -- record that he is also a director and officer of Rocksoft.

MR. FLORES: He is.

THE COURT: But he doesn't sign for Rocksoft, he signs for ACN.

MR. FLORES: It should have said Rocksoft.

THE COURT: It should have.

MR. FLORES: If you read the license --

THE COURT: You are a big law firm. These are big

sophisticated companies, and they come in saying, well, just go with our intent.

MR. FLORES: It's not just intent. With respect, Your Honor, I believe it is supported by the definition of the licensor in the license.

THE COURT: Yeah. And it says and the subsidiaries.

MR. FLORES: And also by the warranty in the license.

THE COURT: What if your neighbor says I hereby sign away on behalf of myself and all of my neighbors all of their houses on this street, you would be the first one to object and say where did they get the authority to sign away my house.

MR. FLORES: You bet I would.

THE COURT: You would. Just because they are related corporations does not give them automatic authority to do that. And just because he happens to be on the boards of both doesn't give him -- you know, I've seen cases where it's the other way, where somebody says -- sometimes comes up in cross-license agreements of what got cross-licensed, and they say, oh, wait, that company never signed.

So sometimes this -- it makes a difference who signs. There is one signature here, and it purports to be on behalf of ACN. So that's problem you got.

MR. FLORES: Well, Your Honor, with respect, we think that the license on its face clearly indicates an intent

to bind Rocksoft by the definition and by the warrants made herein.

THE COURT: It indicates an intent to -- by ACN to do so, true, not an intent by Rocksoft to do so.

MR. FLORES: If that is Your Honor's opinion, then let me point out that the particular facts of this case that I think are really controlling are this: Not only is there the ownership of the stock, but there is composition of the directors and there is the record of how this patent was controlled by Quantum throughout.

The latter point I would refer you to our answer Interrogatory No. 3, which is attached as an exhibit to Mr. Briggs' declaration, I believe it's Exhibit G. And that response interrogatory makes it very clear that since Quantum assumed these rights, it has asserted exclusive control over the patent.

Furthermore, Quantum not only owned all the stock in ACN120, but all of the directors, all of them, were either officers or employees of Quantum. ACN120 not only owned all the stock of Rocksoft, but all the directors of Rocksoft were officers and employees of Quantum and were also directors of ACN. I mean, for all practical purposes, we are not talking about distinct companies. And that is our position.

THE COURT: I think perhaps you can make that argument, but here is the difficulty: If we were to go

forward, and let's say you won or you lost, it almost wouldn't matter. We could build an entire edifice of litigation based upon the assumption that we have standing, and the Federal Circuit can say close the hymnal, stop the choir, sermon over, this case should have never gone to square two. And we would have wasted all that effort. So that is the risk that we are up against if I rule for you, is that we are -- we are forever at risk that this wasn't done right and the I's weren't dotted and the T's weren't crossed.

I see the argument that you are making, I'm not saying it's correct, and I sympathize with those lawyers who with the simple thing of putting in another signature block could have solved this problem then and there. The same guy could have signed for both.

MR. FLORES: We could solve that problem, Your Honor.

THE COURT: No, you can't, you can't do it after the fact.

MR. FLORES: I would refer the Court to the Atmel decision.

THE COURT: I don't agree with Judge Wilken on that. I read that case, I think she ignored Section 261. And I respectfully think that that case is problematic.

MR. FLORES: Your Honor, may I address that point?

THE COURT: Go ahead.

MR. FLORES: I don't think there is any dispute about the distinction between constitutional and prudential standard, the standing. And I don't think there is any dispute about the rule that if you have constitutional standing then you can remedy during the course of the litigation to acquire prudential standing.

I think the Court should directly take a look at the rules for constitutional inquiry. Now, the technicalities about you have to have your I's dotted and your T's crossed, that may apply to completing transfers for acquiring both prudential and constitutional, but the Article 3 test for constitutional injury is a flexible and a practical standard. Focusing on dotting I's and crossing T's is really inappropriate in looking at that issue.

Here, under the facts of this case, given the absolute control that Quantum has over its subsidiaries, given the fact that these directorates are so interlocking and so totally controlled by Quantum, that for all practical purposes they are one in the same company.

You know, this patent, like all other patents, stakes out ownership in some commercial space; there is no doubt that Quantum Corporation controlled that commercial

space. If you read the interrogatory response to Interrogatory No. 3 attached to Mr. Briggs' declaration, it makes that absolutely clear. Since Quantum acquired these rights, it has

exercised control. So the formalities that may govern the first issue I think have no application in the constitutional inquiry.

When there was infringement in this case, given the identity of Quantum with its subsidiaries, when there was an infringement in this case, it was Quantum that got hurt. That gives it constitutional injury in fact under Article 3.

Now, given -- I'm assuming that Your Honor is going to rule that you disagree with us and that the actual transfer of rights by -- on the part of Rocksoft was not effective; even assuming that is the case, there was injury at Quantum, and that, that supports allowing Quantum to remedy any defect in its constitutional -- in its prudential standing during the pendency of this litigation.

Now, Your Honor may disagree with the way those principles were applied by Judge Wilken in the Atmel case, but the fundamental rule that I just articulated I think is nevertheless correct, and it's set forth in many of the cases that are cited. In fact, I don't even think it's really disputed.

So I would urge the Court to confront directly the question of Article 3 standing and to focus on that. I mean, the question there is a flexible one and a practical one. That is very clear from all the Supreme Court decisions, from Warth v. Seldin in the '70s to Medimmune and Genentech just recently.

It's a practical inquiry, not a focus on dotting the I's and crossing the T's.

As a practical matter, these entities, Quantum, ACN120, Rocksoft, are one in the same. And as a practical matter, Quantum has controlled this patent. That should suffice to be an Article 3 injury in fact. And that is why I say that this case should not be decided on broad principles of agency law, it should be decided on federal common law standard, should be decided on flexible practical Article 3

inquiry.

Quantum has a personal stake in this case, not just because it owns a stock of its subsidiaries. Mere stock ownership would not suffice. It has a personal stake in this case because of its absolute control over its subsidiaries to the point that for all practical purposes they are indistinguishable, they are one in the same entity. It has a personal stake because Quantum has absolutely controlled these patent rights, because it is Quantum, as a practical matter, that is being injured by this infringement.

So I don't think Your Honor has to agree with everything in Atmel or how these principles were applied in Atmel, I would urge the Court to just directly look at the test for constitutional injury in fact. And I believe that Quantum Corporation here satisfies it, and that given the opportunity to remedy any -- any technical issues about dotting I's and

crossing T's we'll do so, and we should go forward.

There is no reason for this case to be in Delaware. We've got two California corporations. I believe, with all due respect, that River --

THE COURT: When did the infringement start, the alleged infringement start?

MR. FLORES: That I frankly don't know, Your Honor.

THE COURT: Isn't there a six-year period for Statute of Limitations?

MR. FLORES: As far as damages goes, yes, Your Honor.

THE COURT: Has it been within six years?

MR. FLORES: Yes, Your Honor. And it certainly was occurring at the time the complaint was filed. That much I can tell you.

THE COURT: And if you did have to litigate this in Delaware, you won't even lose any of the period of the Statute of Limitations because you --

MR. FLORES: No.

THE COURT: You could file back there. And so what is the problem in litigating in Delaware?

MR. FLORES: The problem, Your Honor, is first of

all, it's in Delaware, and there is no reason to seek an inconvenient forum. Delaware, to me, makes no sense.

THE COURT: Inconvenient to them, too, I guess.

MR. FLORES: It would -- yes.

THE COURT: So what is really turning on -- I mean, I don't understand the strategies here. What really turns -- even if it is inconvenient, what difference does it make whether it's back there or here?

MR. FLORES: Delay, Your Honor, delay. I mean, I -- this, Your Honor, will push this case forward, I know that, the parties know that. We are going to proceed efficiently and expeditiously to trial because you are going to require that. We go to Delaware, we are going to start again from square one. It's just going to be more delay and more expense.

That is really the only effect of going to Delaware. This is -- that's why I believe it's the only reason I can think of that Riverbed would file this motion. And I think that under the Article 3, under the flexible practical standards the Supreme Court has always applied, there has clearly been an injury to Quantum, and Quantum should be allowed to remedy that. And we should stay in this case before Your Honor and proceed efficiently, taking advantage of all the work we've done.

As you pointed out, these cases are difficult and they are technical. Packing up and moving it to Delaware is going to be a lot of extra unnecessary expense and, even more important, a lot of unnecessary delay.

THE COURT: Well, one possibility is that I could

just stay this entire case until the appeal got resolved. Then nobody's case would go forward.

MR. FLORES: Your Honor, that is not a possibility that Quantum wants to take advantage of. We want to move forward with the litigation. We want to move forward in the most expeditious way possible. Quantum is not interested in appeals.

The other possibility Your Honor raised is to give

us leave to file a counter-claim to the counter-claim.
THE COURT: Well, I don't know if that is authorized by the Rules, but I'm not saying it's not. But if it is authorized by the Rules, then maybe there is -- maybe then you just do that, and we are back to square one.
MR. FLORES: Well --
THE COURT: Except your damages period would be a little different. Probably doesn't even matter.
MR. FLORES: We would be back to square one, in the best sense of the word, because we would be back on this Court's schedule, which we want to adhere to.
THE COURT: All right, so sounds like what matters here is the schedule. You want -- probably what's going on here is this side of the room knows that I will move the case along, so they want their patent to be the one, they want their offensive case to be the one that moves along and have your offense case mired down in some other court somewhere else in

the country. That is probably what is happening here.
And I don't mean any disrespect to any other court in the country, not at all. Everybody has to manage their caseload the way they want, but sounds like that's what's happening here.
MR. FLORES: With respect to my colleagues, I believe that to be the case.
THE COURT: All right. Thank you.
Mr. Briggs, do you want to respond?
MR. BRIGGS: Sure. Thank you.
He talked a lot about the constitutional injury --
THE COURT: You should refer to him by name.
MR. BRIGGS: Mr. Flores as the constitutional --
THE COURT: One of the things about oral arguments you must be exceedingly courteous to your opponent.
MR. BRIGGS: Thank you.
THE COURT: He has a name, refer to him by name.
MR. BRIGGS: Mr. Flores talked a lot about the constitutional injury point, and I would just like to direct the Court's attention to a Federal Circuit case called Schreiber. Now, Schreiber completely undercuts their argument

that they had constitutional injury as a parent corporation. In that case, the parent filed suit, and it owned the patent at that time. And during the course of the lawsuit, it assigned the patent to a subsidiary. And later on in the lawsuit, the

patent was transferred back to the parent.

But for the brief period of time when the parent had assigned the patent to its subsidiary, the Federal Circuit held that it had lost its complete stake in the -- in the action and did not have constitutional standing. So that -- that case undermines his argument. It undermines the Atmel case as well.

There is another District Court case where Judge Posner was sitting by designation that is also very interesting to constitutional injury point. It is DePuy versus Zimmer Holdings. And that was the United States District Court in Illinois, 2005, 384 F. Supp. 2d. And he had a very similar issue to what is going on here.

In that case, the patent at issue was owned by a wholly-owned subsidiary of DePuy, and the parent argued that since it owns the patentee, it suffered constitutional injury. And there is a very interesting analysis done by Judge Posner in that case, and he found that there was no constitutional injury by the parent.

The third point I want to make here is that if the transfer on August 13th from ACN120 to Quantum actually did transfer rights in the patent, that -- that is -- is completely undermined by the December 24th assignment from -- from Rocksoft to ACN120. Why would they need that assignment if the patent had already been transferred?

THE COURT: Well, belt and suspenders. Insurance.

I think it recognizes there is an issue, but it doesn't necessarily mean they were wrong the first time. It draws into question whether they were wrong, but it doesn't conclusively prove they were wrong.

MR. BRIGGS: Right.

Well, there is also language in that agreement that states that they hadn't previously transferred any rights that are inconsistent with that agreement. So at that point in

time, they were saying that we -- Rocksoft said that it had not previously transferred any rights inconsistent with that agreement.

THE COURT: Okay.

Okay, here is what I want you to do, you should send a letter or just make a filing by noon tomorrow, telling the Court whether -- if this motion is granted the Court can also dismiss your counter-claim; in other words, it would be deemed voluntarily withdrawn such that there would be nothing left in this court.

Now, I'm not saying that's a condition of granting your motion, but I would like to know the answer to that before I issue a ruling.

MR. BRIGGS: Okay.

THE COURT: So you make whatever call you want on that, but ifs, but and maybes won't do any good. So it's got to be yes, we will, or no, we won't.

All right, I want to thank you everybody for their efforts here. And we'll get an order out probably on Monday or Tuesday, okay?

MR. BRIGGS: Thank you, Your Honor.

MR. STERN: Thank you very much, Your Honor.

THE COURT: All right.

MR. FLORES: Thank you, Your Honor.

(Proceedings adjourned at 8:55 a.m.)

---o0o---

22

23

24

25

CERTIFICATE OF REPORTER

I, Sahar McVickar, Official Court Reporter for the United States Court, Northern District of California, hereby certify that the foregoing proceedings were reported by me, a certified shorthand reporter, and were thereafter transcribed under my direction into typewriting; that the foregoing is a full, complete and true record of said proceedings as bound by me at the time of filing. The validity of the reporter's certification of said transcript may be void upon disassembly and/or removal from the court file.

Sahar McVickar, RPR, CSR No. 12963

February 5, 2008

EXHIBIT F

quinn emanuel trial lawyers | silicon valley
555 Twin Dolphin Drive, Suite 560, Redwood Shores, California 94065 | TEL: (650) 801-5000 FAX: (650) 801-5100

February 1, 2008

VIA ELECTRONIC FILING

Honorable William Alsup
450 Golden Gate Ave.
Courtroom 9, 19th Floor
San Francisco, CA 94102

Re: *Quantum Corporation v. Riverbed Technology, Inc.* (No. C 07-4161 WHA) – Court's Inquiry Regarding Riverbed's Patent Infringement Counterclaim

Dear Judge Alsup:

In furtherance of Your Honor's instruction at the hearing yesterday on Riverbed's motion to dismiss Quantum's complaint for lack of standing, Riverbed informs the Court that, should this Court grant Riverbed's motion, Riverbed would be inclined to maintain its patent infringement counterclaim under U.S. Patent No. 7,116,249 against Quantum in this Court.

Respectfully submitted,

QUINN EMANUEL URQUHART OLIVER & HEDGES, LLP

Claude M. Stern
Counsel for Defendant and Counterclaimant, Riverbed Technology, Inc.

EXHIBIT G

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

QUANTUM CORPORATION,

Plaintiff,

v.

RIVERBED TECHNOLOGY, INC.,

Defendant.
_____/

AND RELATED COUNTERCLAIMS.
_____/

No. C 07-04161 WHA

ORDER GRANTING DEFENDANT'S MOTION TO DISMISS

INTRODUCTION

In this patent-infringement action, defendant moves to dismiss, contending plaintiff lacked any rights to the patent asserted at the time the complaint was filed. For the reasons stated below, defendant's motion is **GRANTED**.

STATEMENT

Plaintiff filed its complaint on August 14, 2007, alleging infringement of U.S. Patent No. 5,990,810, which concerns a method for data processing and storage. The inventor assigned his rights in the '810 patent to Trustus Pty., Ltd., which assigned all rights to Rocksoft Ltd. (Bruno Decl. Exh. 1). Advanced Digital Information Corp. then formed and registered Australian Company Number 120 786 012 Pty. Ltd., ("A.C.N. 120") as a proprietary company, both Rocksoft and A.C.N. 120 being registered in Australia (Hall Decl. Exh. 5). A.C.N. 120

then acquired all the shares of Rocksoft, such that Rocksoft became a wholly-owned subsidiary of A.C.N. 120 (*id.* at Exhs. 2–4). Plaintiff Quantum, a U.S. company, then acquired ADIC as a wholly-owned subsidiary (*id.* at Exh. 8). Pursuant to a technology-licensing agreement signed the day before this litigation commenced, A.C.N. 120 "and its subsidiaries" granted an exclusive license to Quantum to "all patents, patent applications, [and] intellectual property rights" owned by A.C.N. 120 and its subsidiaries. Significantly, however, Rocksoft did *not* sign the agreement. On December 24, 2007, four months after the complaint was filed, Rocksoft executed an assignment of the "entire right, title and interest" in the '810 patent to A.C.N. 120, although Quantum was not a party (Briggs Decl. Exh. H). Defendant now moves to dismiss this action under FRCP 12(b)(1), claiming plaintiff had insufficient interest in the '810 patent at the time the complaint was filed.

ANALYSIS

1. LEGAL STANDARD.

Standing of the parties to bring their claims is a threshold matter that must be addressed before the substantive merits of the case. A patent plaintiff must have a "personal stake" in a case so as to assure that any harm may be redressed by the outcome of litigation. *Stoianoff v. State of Mont.*, 695 F.2d 1214, 1223 (9th Cir. 1983). In the patent infringement context, "an exclusive licensee has standing to sue in its own name, without joining the patent holder, where all substantial rights in the patent are transferred." *International Gamco, Inc. v. Multimedia Games, Inc.*, 504 F.3d 1273, 1276 (Fed. Cir. 2007). In such a case, the exclusive licensee is effectively an assignee. "Applications for patent, patents, or any interest therein, shall be assignable in law by an instrument in writing." 35 U.S.C. 261. Where standing to sue for patent infringement depends on a written instrument, it must be executed before the filing of the complaint. *See Enzo APA & Son, Inc. v. Geapag AG*, 134 F.3d 1090, 1093–94 (Fed. Cir. 1998);

2. PLAINTIFF HAS NO STANDING.

It is undisputed that Rocksoft held proper title to the patent pursuant to the November 2004 licensing agreement. Defendant contends that as of the filing date of the complaint, however, plaintiff had no rights in the '810 patent because the chain of title ended

at Rocksoft, meaning there was no assignment linking Rocksoft to A.C.N. 120 or beyond. In reply, plaintiff argues that Quantum acquired its interest in the patent pursuant to the licensing agreement entered between A.C.N. 120 and Quantum one day before the complaint was filed. It provided:

> Licensor hereby grants to Licensee and Licensee hereby accepts, a perpetual, exclusive, world-wide, transferable license with the right to sublicense Licensor's entire rights under the Licensed Technology, including without limitation, the right to manufacture, make, have made, use, import, export, sell and service products that incorporate the Licensed Technology (Briggs Decl. Exh. 2).

The agreement defined "Licensed Technology" as "all patents . . . that Licensor owns or licences" and "Licensor" as "A.C.N. 120 . . . and its subsidiaries" (*ibid.*).

Trouble is, Rocksoft was not a party to the agreement, *i.e.*, Rocksoft did not sign the agreement. The question thus becomes whether, absent an alter ego showing, a parent company is automatically deemed to be the agent of its subsidiaries authorized to transfer patent rights.

The record does not come close to showing that A.C.N. 120 should be treated as Rocksoft's alter ego.* As the court in *Aladdin Oil Corp. v. Perluss*, 230 Cal. App. 2d 603, 614 (1965) held:

> Parties who determine to avail themselves of the right to do business by means of the establishment of a corporate entity must assume the burdens thereof as well as the privileges. The alter ego doctrine is applied to avoid inequitable results not to eliminate the consequences of corporate operations.

While Rocksoft and A.C.N. 120 were affiliates, each was created as separate corporate entities, and must accordingly be treated as such. That is, of course, the whole point of a corporation, to isolate its assets, liabilities, and operations. Its assets can be alienated only by the officers and directors named in the charter and bylaws or, of course, through its authorized agents. It is true that A.C.N. 120 purported to act on behalf of its subsidiaries, but nothing in the record shows that A.C.N. 120 was actually authorized to act as an agent for Rocksoft, with respect to the patent in suit or any other asset. Contrary to plaintiff, the mere fact that Rocksoft was A.C.N. 120's

---

* At the argument, counsel claimed the companies are "practically one and the same. *This* is only argument. Corporate control is not enough by itself to prove alter ego. No attempt has been made to prove alter ego. Also, no caselaw has been provided that an alter-ego showing would satisfy the written requirements of Section 261.

wholly-owned subsidiary does not automatically mean that A.C.N. 120 and Rocksoft had an agency relationship. As the Supreme Court held in *Dole Food Co. v. Patrickson*, 538 U.S. 468, 474–75 (2003):

> A basic tenet of American corporate law is that the corporation and its shareholders are distinct entities. An individual shareholder, by virtue of his ownership of shares, does not own the corporation's assets and, as a result, does not own subsidiary corporations in which the corporation holds an interest. A corporate parent which owns the shares of a subsidiary does not, for that reason alone, own or have legal title to the assets of the subsidiary; and, it follows with even greater force, the parent does not own or have legal title to the subsidiaries of the subsidiary. (internal citations omitted)

In addition, in *Schreiber Foods v. Beatrice Cheese*, Inc., 402 F.3d 1198 (Fed. Cir. 2005), the parent corporation, Schreiber, assigned its rights to a patent during a lawsuit to its subsidiary, Schreiber Technologies, which in turn granted Schreiber a non-exclusive license to the patent. The assignment was apparently done for tax purposes. Despite the fact that Schreiber still necessarily controlled the patent through its subsidiary, the Federal Circuit found that Schreiber had lost standing, holding (*id.* at 1202-03):

> [O]nce the assignment to Schreiber Technologies was completed, there was no question that Schreiber lost its 'personal stake in the outcome.'

Similarly, Quantum's control of its subsidiaries is insufficient to confer standing.

The later assignment in December 2007, four months after the complaint was filed, suggests that plaintiff realized that the August 2007 licensing agreement needed fixing. Plaintiff now contends (Opp. 2–3):

> The [December 2007] assignment was designed to show a clear chain of title on record in the [USPTO] so as to avoid any questions in the future by parties who have not had the benefit of reviewing the terms of Quantum's exclusive license.

This order assumes arguendo that the August 2007 licensing agreement in combination with the December 2007 assignment was enough as of the later date. Nonetheless, the chain of title was not perfected when the lawsuit began, a critical distinction under Federal Circuit law. *See Paradise Creations, Inc. v. UV Sales, Inc.*, 315 F.3d 1304, 1310 (Fed. Cir. 2003).

Atmel Corp. v. Authentec, Inc., 490 F. Supp. 2d 1052 (N.D. Cal. 2007), does not save the day for plaintiff. In *Atmel*, Judge Claudia Wilken of this district allowed a parent corporation, Atmel Corporation, to assert a patent jointly owned by its subsidiaries, Atmel Grenoble and Atmel Switzerland, despite the absence of an express licensing agreement between the corporations. This order, however, did not come to grips with the law cited above, perhaps because it was not adequately presented. Under 35 U.S.C. 261, to repeat, "[a]pplications for patent, patents, or any interest therein, shall be assignable in law by an instrument in writing," a point of law perhaps not in contention in that litigation. As the Federal Circuit in *Enzo APA*, 134 F.3d at 1093, held (emphasis added):

> While we acknowledge that a license may be written, verbal, or implied, if the license is to be considered a virtual assignment to assert standing, it *must be in writing*. . . . As such, the licensing arrangement conferring such must, logically, resemble an assignment in both form and substance.

Given that the August 2007 licensing agreement was not signed by Rocksoft, no rights in the '810 patent were transferred by that agreement. As no other written assignment or license agreement was executed before the filing of the complaint, *Atmel* is distinguishable.

* * *

In light of the proliferation of patent-infringement actions, it is not too much to ask sophisticated patent litigants to be careful when it comes to the threshold issue of standing. It is a simple task to execute express license agreements that satisfy the Federal Circuit standard. Among affiliated companies, it should be even simpler. It is true that patent litigants sometimes rush to stake out venue in a preferred forum. A rush to sue, however, cannot excuse the stern necessity of perfecting the required title before suit. District judges cannot overlook a defect in the chain of title, for the entirety of massive litigation might wind up being vacated years later, for lack of threshold standing. *See Gaia Techs., Inc., v. Reconversion Techs., Inc.*, 93 F.3d 774 (Fed. Cir. 1996) (vacating a final judgment after a full trial on the merits because of a deficiency in standing). As carpenters say, it is wise to "measure twice and cut once."

It is worth noting that very little prejudice will flow from this ruling. There is a parallel suit anchored in Delaware between the very same parties and over the very same patent. The

damages claims may be asserted there. Conceivably, it could even be asserted as a counterclaim to the counterclaim in the instant action, although that type of procedural maneuver would need to be validated. In either event, plaintiff should double check its assumption that the later assignment cured any defect in Quantum's standing.

3. PLAINTIFF'S ADMINISTRATIVE MOTION TO FILE SUPPLEMENTAL DECLARATION.

At the eleventh hour, plaintiff requested leave to supplement the record with evidence of A.C.N. 120's authority to sign on behalf of Rocksoft. Plaintiff's request, however, was made after all briefing was completed. Plaintiff contends that such evidence should be considered because for the first time in its reply, Riverbed addressed the issue of whether the August 2007 agreement included Rocksoft. It was clear from the defendant's motion, however, that Riverbed was directly challenging whether or not the agreement between A.C.N. 120 and Quantum transferred any rights in the patent owned by Rocksoft. That was in fact the *only* issue presented by defendant in its motion. In its opposition, plaintiff argued that the August 2007 license agreement properly transferred the patent rights. Plaintiff should have easily anticipated that any evidence relating to A.C.N. 120's authority to bind Rocksoft would have easily fell within the ambit of its opposition to this motion. Accordingly, plaintiff's administrative request to supplement the record is **DENIED**.

It is worth noting that the proffer is insufficient anyway. The decisive fact is that no signature for Rocksoft was on the August 2007 agreement. Even if the signor for A.C.N. 120 was also an officer of Rocksoft and could have also signed for Rocksoft, there was no such signature block on the signature page. Only A.C.N. 120 purported to sign.

CONCLUSION

For the foregoing reasons, defendant's motion to dismiss is **GRANTED**.

IT IS SO ORDERED.

Dated: February 4, 2008.

WILLIAM ALSUP
UNITED STATES DISTRICT JUDGE

EXHIBIT H

TECHNOLOGY AND TRADEMARK LICENSE AGREEEMENT

This Agreement ("Agreement") is made effective as of August 22, 2006 ("Effective Date") by and between Rocksoft Pty Limited (ABN 47 008 280 153), a company registered in South Australia and incorporated under the laws of Australia, with its registered office at Level 4, Deutsche Bank Place, Corner Hunter and Phillip Streets, Sydney, New South Wales 2000, Australia ("Licensor") and Quantum Corporation ("Licensee"), a Delaware corporation with offices at 1650 Technology Drive, Suite 700, San Jose, CA 95110, U.S.A.

RECITALS

WHEREAS, the Licensor owns the rights in respect of the Licensed Marketing Intangibles and the Licensed Technology, including, without limitation, US Patent No 5,990,810 ("the '810 patent"), which concerns a method for data processing and storage.

WHEREAS, the inventor assigned his rights in the '810 patent to Trustus Pty Limited, which in turn assigned all its rights to the Licensor, a wholly-owned a subsidiary of ACN 120 786 012 Pty Ltd.

WHEREAS, ACN 120 786 012 Pty Ltd and the Licensee entered into the Technology and Trademark License Agreement ("License Agreement") on or about 13 August 2007 (expressed to be effective as of 22 August 2006), to grant a perpetual exclusive world-wide license in respect of the Licensed Marketing Intangibles and the Licensed Technology, which included the '810 patent.

WHEREAS, the United States District Court for the Northern District of California in proceedings number C07-04161 WHA between the Licensee and Riverbed Technology, Inc., found in its reasons for granting Riverbed's Motion to Dismiss, dated February 4 2008 (by US District Judge William Alsup) that the License Agreement was ineffective to transfer rights in the '810 patent.

WHEREAS, the parties acknowledge that notwithstanding the findings of US District Judge William Alsup, to the extent the License Agreement was effective to transfer any relevant rights, they wish to terminate the License Agreement with effect from February 11, 2008 (the "Effective Termination Date").

WHEREAS, in accordance with Article 8.1 of the License Agreement, by mutual agreement between ACN 120 786 012 Pty Ltd and the Licensee under the Termination of Technology and Trademark Licensing Agreement, those parties agreed that the License Agreement shall terminate with effect from the Effective Termination Date.

WHEREAS, the parties now wish to enter into an effective licensing arrangement in respect of the rights in the '810 patent in favour of the Licensee.

WHEREAS, Licensor warrants that it owns the Licensed Technology and Licensed

Marketing Intangibles defined herein.

WHEREAS, Licensor desires to grant and Licensee desires to acquire a perpetual exclusive worldwide license with the right to sublicense to the Licensed Technology and the Licensed Marketing Intangibles and whereas Licensee agrees to compensate Licensor for this license on the terms and conditions set forth herein.

WHEREAS, the rights licensed under this Agreement include all rights and benefits relating to the Licensed Technology including, without limitation, the right of the Licensee to bring action and claim relief in respect of any infringement or unauthorised use of the Licensed Technology whether occurring before, on, or after, the date of this Agreement.

NOW THEREFORE, in consideration of the foregoing recitals and the mutual covenants and conditions set forth herein, the parties agree as follows:

ARTICLE 1 – DEFINITIONS

1.1 "Licensed Marketing Intangibles" means any trademarks, service marks, trade names, brand names, copyrights, designs, logos, symbols, customer lists, packaging and similar property in existence on the Effective Date, or thereafter, that Licensor owns or licenses. Licensee acknowledges that the nature and quality of the goods or services sold under this Agreement are subject to the control of the Licensor. Licensee agrees to comply with all quality control provisions provided by Licensor.

1.2 "Licensed Technology" means all patents, patent applications, proprietary rights, intellectual property rights, inventions, copyrights, computer programs, mask works, software, source code, enhancements, updates, translations, adaptations, secret and confidential information, data, specifications, designs, process technology, know-how and other intangible property (whether or not patentable or copyrightable) in existence on the Effective Date, or thereafter, that Licensor owns or licenses including, without limitation, United States Patent No 5,990,810 and any other letters patent that might be granted for the invention in respect of that patent in the United States and throughout the world.

ARTICLE 2 – GRANT OF TECHNOLOGY LICENSE

2.1 Subject to the terms and conditions in this Agreement, Licensor hereby grants to Licensee,

and Licensee hereby accepts, a perpetual, exclusive, world-wide, transferable license with the right to sublicense or assign all or any part of Licensor's rights under the Licensed Technology, including without limitation, the right to manufacture, make, have made, use, import, export, sell and service products that incorporate the Licensed Technology.

2.2 Subject to the terms and conditions in this Agreement, Licensor hereby grants to Licensee, and Licensee hereby accepts, a perpetual, exclusive, world-wide, transferable license with the right to sublicense or assign all or any part of Licensor's rights under the Licensed Marketing Intangibles, including without limitation, the right to reproduce, distribute, and use the Licensed Marketing Intangibles.

2.3 The license granted by this Agreement extends to the Licensee's use of the Licensed Marketing Intangibles and the Licensed Technology in existence on the Effective Date throughout the life of the Licensed Technology, regardless of when such use occurred.

ARTICLE 3 – CONSIDERATION

3.1 Royalty Payments. In consideration for the rights granted to Licensee hereunder, Licensee shall pay to Licensor royalty payments in an amount to be determined by the parties based on an arms-length report to be agreed upon by the parties which amount shall be, if not agreed otherwise, the amount most recently paid under the now terminated License Agreement. Royalty payments shall be made in U.S. dollars and shall be made quarterly or when requested.

ARTICLE 4 – DELIVERY OF TECHNOLOGY

4.1 During the term of this Agreement, Licensor shall, when requested by Licensee, deliver to Licensee, or grant Licensee access to, the Licensed Technology for its or Licensee's designated third party's use.

ARTICLE 5- LICENSOR'S & LICENSEE'S REPRESENTATIONS AND WARRANTIES

5.1 Licensor represents and warrants to Licensee that as of the date of delivery, the Licensed Technology made available to the Licensee hereunder will be the complete and current version. Licensor represents and warrants that U.S. Patent No. 5,990,810 and all other Licensed Technology is in force.

5.2 Indemnification for Infringement. Licensor shall defend, indemnify and hold Licensee harmless against any and all claims, demands, suits, proceedings, losses, liabilities, damages, costs and expenses (including reasonable attorneys' fees) (collectively "Liabilities") which are attributable to any allegation that the Licensed Technology infringes any other person's, firm's or entity's patent, copyright, trademark, trade secret or other proprietary right. Licensor shall pay any costs and damages awarded against Licensee that are attributable to any such claim. Licensee will promptly notify Licensor of any claims, indications or objections that the use of the Licensed Technology may or will infringe the intellectual property or proprietary rights of any third party.

5.3 The Licensor represents and warrants to the Licensee that:

(a) it is a body corporate validly existing under the laws of its place of incorporation or establishment;

(b) it has the corporate power to enter into and perform its obligations under this Agreement and to carry out the transactions contemplated by this Agreement;

(c) it has taken all necessary corporate action to authorise the entry into and performance of this Agreement and to carry out the transactions contemplated by this Agreement; and

(d) this Agreement is a valid and binding obligation in accordance with its terms and conditions.

5.4 The Licensee represents and warrants to the Licensor that:

(a) it is a body corporate validly existing under the laws of its place of incorporation or establishment;

(b) it has the corporate power to enter into and perform its obligations under this Agreement and to carry out the transactions contemplated by this Agreement;

(c) it has taken all necessary corporate action to authorise the entry into and performance of this Agreement and to carry out the transactions contemplated by this Agreement; and

(d) this Agreement is a valid and binding obligation in accordance with its terms and conditions.

ARTICLE 6 – CONFIDENTIAL INFORMATION

6.1 Licensor and Licensee agree that the provisions of this Agreement shall be maintained in confidence and shall not be disclosed to any third party absent written consent from the non-disclosing party to this Agreement.

ARTICLE 7 - INTELLECTUAL PROPERTY RIGHTS

7.1 Ownership. Licensor warrants that Licensor is, and shall remain, the exclusive owner of all rights, title and interest in and to, or the authorized licensee from a third party of, the Licensed Technology and the Licensed Marketing Intangibles. Licensee shall acquire no rights whatsoever in or to any of the Licensed Technology and the Licensed Marketing Intangibles, except as specifically provided for in this Agreement.

7.2 Maintenance and Protection of Intellectual Property Rights. Licensee has the exclusive right to preserve the validity and enforceability of all rights, title and interest in and to the Licensed Marketing Intangibles and the Licensed Technology, including, without limitation, the right to sue for infringement occurring in the future, past or present, at Licensee's sole discretion. The Licensee may retain the benefit of any damages or other relief that may arise from such proceedings brought by the Licensee. Licensor shall provide Licensee with such assistance as Licensee shall reasonably request in connection with preserving such validity and enforceability. Licensor agrees that it shall promptly notify Licensee of any and all infringements, imitations, illegal use, or misuse by any person, firm or entity of the Licensed Marketing Intangibles and the Licensed Technology which come to its attention. Licensor shall promptly notify Licensee in writing of all maintenance fee due dates, renewal due dates, and other statutory or regulatory deadlines for taking steps that may preserve or enhance the intellectual property rights licensed hereunder.

ARTICLE 8 - DURATION AND TERMINATION

8.1 The term of this Agreement shall commence on the date first written above and shall remain in force and effect in perpetuity unless terminated: (a) by mutual agreement between the parties hereto, or (b) or in accordance with the provisions of this Article.

8.2 Licensee, in its sole discretion, may terminate this Agreement for any reason at any time by giving thirty (30) days prior written notice to Licensor. Licensee may terminate the entire agreement, or the agreement only as to specified Licensed Technology and/or Licensed Marketing Intangibles.

ARTICLE 9 – EFFECT OF TERMINATION

9.1 Upon termination of this Agreement, Licensee or Licensee's designated third parties shall have the right to liquidate its existing inventory of product incorporating the Licensed Technology.

9.2 The provisions of Articles 1, 5 – 7 and 9 – 13 survive termination.

ARTICLE 10 - COMPLIANCE WITH LAWS

10.1 General Compliance. Each party shall at all times and at its own expense (1) strictly comply with all applicable laws, rules, regulations and governmental orders, now or hereafter in effect, relating to its performance of this Agreement; (2) pay all fees and other charges required by such laws, rules, regulations and orders; and (3) maintain in full force and effect all licenses, permits, authorizations, registrations and qualifications from all applicable governmental departments and agencies to the extent necessary to perform its obligations hereunder.

10.2 U.S. Export Controls. Without limiting the generality of Article 10.1 hereof, Licensee specifically acknowledges that certain of the Licensed Technology ("Technical Data") is subject to United States export controls, pursuant to the Export Administration Regulations, 15 C.F.R. Parts 768-799. The parties shall comply strictly with all requirements of the Export Administration Regulations with respect to all such Technical Data. Without limiting the generality of the foregoing obligation, the parties hereby expressly agrees that without the prior written authorization of the United States Commerce Department, the parties will not, and will cause their respective representatives to agree not to (a) export, reexport, divert or transfer any such Technical Data, or any direct product thereof, to any destination, company, or person prohibited by the Export Administration Regulations, including the Table of Denial Orders, or (b) disclose any such Technical Data to any national of any country when such disclosure is prohibited by the Export Administration Regulations.

ARTICLE 11 – ASSIGNMENT

11.1 Licensee may assign, delegate or transfer all or any part of its rights or obligations under this Agreement at its sole discretion and without obtaining Licensor's prior consent.

11.2 This Agreement shall inure to the benefit of, and be binding upon, each of the parties, their successors and permitted assignees, delegates or transferees.

ARTICLE 12 – GOVERNING LAW AND DISPUTE RESOLUTION

12.1 Choice of Law. This Agreement is governed by the laws of New South Wales. Each party submits to the jurisdiction of the courts exercising jurisdiction there, and waives any right to claim that those courts are an inconvenient forum.

12.2 Legal Expenses. Subject to any order of a court, the prevailing party in any legal proceeding brought by one party against the other arising out of this Agreement shall be entitled to recover its legal expenses, including court costs and reasonable attorney's fees.

ARTICLE 13 – GENERAL PROVISIONS

13.1 No Waivers. The failure of either party to assert any of its rights under this Agreement shall not be deemed to constitute a waiver of that party's right thereafter to enforce every other provision of this Agreement in accordance with its terms.

13.2 Entire Agreement. This Agreement, including any schedules attached hereto, constitutes the entire agreement of the parties concerning the use of the Licensed Technology and supersedes all prior agreements, understandings and communications, whether written or oral, between the parties with respect to the subject matter hereof. No modification or amendment of this Agreement shall be effective unless in writing and executed by a duly authorized representative of each party. The parties agree to cooperate to supplement, modify, and amend the understandings in this Agreement if circumstances make it appropriate to do so.

13.3 Force Majeure. Notwithstanding anything in this Agreement to the contrary, neither party shall be liable to the other party for any failure to perform or delay in the performance of any obligation hereunder, other than an obligation to pay money, when such failure to perform or delay in performance is caused by an event of force majeure: provided, however, that the party whose performance is prevented or delayed by such event of force majeure shall give prompt notice thereof to the other party. For purposes of this Article, the term force majeure " shall include war, rebellion, civil disturbance, earthquake, fire, flood, strike, lockout, labor unrest, acts of governmental authorities, shortage of materials, acts of God, acts of the public enemy and, in general, any other causes or conditions beyond the reasonable control of the parties.

13.4 Notices. All notices and communications required to be given under this Agreement shall be sent to the respective parties as follows:

a. If to Licensee:

Quantum Corporation

1650 Technology Drive,
Suite 700
San Jose, CA 95110
Facsimile: (408) 944-6581
Attention: General Counsel

b. If to Licensor:

Rocksoft Pty Limited
Level 7, Shell House
~~170 North Terrace~~
Adelaide
South Australia 5000
Facsimile: + 61 7 3853 9401
Attention: Finance Manager

13.5 Notices shall be sent in the following manner:

a. By registered airmail return receipt requested and postage prepaid, or by courier or hand delivery;

b. By telefax, with a copy sent within three days thereafter by registered airmail.

13.6 All such notices, reports, statements and other communications shall be deemed to have been received:

a. If sent by registered, first-class airmail, upon delivery to the addressee and

b. If sent by courier or hand delivery, upon delivery to the addressee.

c. If sent by telefax, upon confirmation obtained by the sender of the receipt of such telefax by the recipient.

13.7 Subject Headings. The subject headings of this Agreement are included for purposes of convenience only and shall not affect the construction or interpretation of any of its provisions.

IN WITNESS HEREOF, the parties have caused this Agreement to be executed by their duly authorized officers.

Quantum Corporation
By:
Title: Jon Gacek, EVP, CFO
Date: 2/8/2008

Executed in accordance with section 127 of the *Corporations Act 2001* by **Rocksoft Pty Limited:**

Director Signature	Director/Secretary Signature
Jon Gacek	**Shawn Hall**
Print Name	Print Name

Date: 2/8/2008